ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э.Баумана, ОАО "Концерн "Сириус".

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru E-mail: prin@novtex.ru

Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 004.031.43:004.272.43:004.451.42

А. Н. Докучаев, аспирант, Балтийский государственный
технический университет "Военмех" имени Д. Ф. Устинова, г. Санкт-Петербург
e-mail: a.n.dokuchaev@gmail.com

Ê îöåíêå ýôôåêòèâíîñòè ìåõàíèçìîâ äèñïåò÷åðèçàöèè ìóëüòèïðîöåññîðíûõ ñèñòåì ðåàëüíîãî âðåìåíè ïðè ó÷åòå âëèÿíèÿ äëèòåëüíûõ áëîêèðîâîê

Рассмотрено влияние длительных блокировок на диспетчеризацию задач в мультипроцессорных системах реального времени при использовании метода, основанного на принудительных вытеснениях. Исследуется возможность применения указанного метода в условиях наличия межзадачного взаимодействия.

Ключевые слова: *системы реального времени, параллелизм, мультипроцессорная диспетчеризация, длительные блокировки, планирование с принудительными периодическими вытеснениями*

Введение

Актуальность применения мультипроцессорного и многоядерного аппаратного обеспечения при проектировании систем реального времени (СРВ) обусловила проведение большого числа исследований, направленных на поиск эффективных методов распределения вычислительных ресурсов между потребителями (задачами). Существует два основных направления подобных исследований. Первое направление заключается непосредственно в выработке эффективных дисциплин диспетчеризации, позволяющих гарантировать своевременность выполнения задач. Как показывает практика, при стремлении суммарной вычислительной нагрузки, необходимой для успешного исполнения набора задач, к предельному значению, соблюдение данного требования оказывается крайне затруднительным, а порой и невозможным. Как следствие, появляется необходимость в исследованиях на втором направлении, которое связано с поиском методов анализа выполнимости задач, составляющих программное обеспечение СРВ. Очевидно, что эти методы не носят универсальный характер, поскольку применимы для строго определенного подмножества дисциплин диспетчеризации.

Постановка задачи

Согласно работам [1, 2] любая периодическая задача τ_i характеризуется в первую очередь следующими параметрами: T_i — периодом порождения экземпляров задачи, C_i — объемом необходимой в периоде вычислительной нагрузки и D_i — крайним сроком завершения (в данной работе рассматривается лишь случай $T_i = D_i$). Авторами работы [1] предложен алгоритм диспетчеризации RMS (*Rate Monotonic Scheduling*), являющийся оптимальным в своем классе для систем с единственным процессором. Под оптимальностью алгоритма планирования понимается способность обеспечивать успешное выполнение (соблюдение крайних сроков) всех задач в любом выполнимом наборе. Крайним сроком завершения объекта диспетчеризации D_i называется относительный момент времени, отсчитываемый от начала периода, до истечения которого задача должна завершиться (потребить весь необходимый ей объем процессорного времени C_i). Для систем "жесткого реального времени", в отличие от систем "мягкого реального времени", превышение крайнего срока завершения хотя бы одной задачей трактуется как неспособность СРВ выполнять свои функции. В качестве механизма регулирования количества потребляемых ресурсов задачами такой алго-

ритм использует понятие приоритета. Основная идея, положенная в основу метода, формулируется в следующем: назначаемый задаче τ_i приоритет тем выше, чем меньше ее период T_i.

В последующих работах было показано, что алгоритм RMS не является оптимальным для задач, выполняющихся в мультипроцессорных СРВ [3]. С этих позиций актуальным становится применение методов, описанных в работах [4, 5]: RM-US и SM-US. Указанные методы используют подход, заключающийся в наделении некоторого подмножества задач исключительным приоритетным преимуществом. Высокими приоритетами наделяются задачи, которым требуется предоставление вычислительных ресурсов, превышающих пороговое значение, определенное авторами. Подобные высокоприоритетные задачи принято именовать тяжелыми. Очевидно, данное решение основано на преднамеренном ущемлении низкоприоритетных (легких) задач в ресурсах процессора.

Имеет смысл предположить, что при существенном числе так называемых тяжелых задач в наборе, некоторым легким задачам не будет предоставлено процессорное время в любом периоде их выполнения. Данная идея легла в основу принципа, именуемого принудительными периодическими вытеснениями. Методами, реализующими данный подход, являются RMMS-FP/SMMS-FP (*Rate/Slack — Monotonic Multicore Scheduling with Forced Preemptions*) [6]. В работе [6] было показано, что наделение тяжелых задач способностью периодически вытесняться (это ключевое отличие указанных алгоритмов от RM-US и SM-US) приводит к увеличению числа переключений контекстов, однако благоприятно сказывается на итоговой эффективности алгоритма диспетчеризации. Тем не менее вопрос о применимости принципа принуждения тяжелых задач к вытеснению оказывается весьма актуальным при исключении требования независимости задач в наборе. Данный вопрос будет детально рассмотрен в настоящей статье.

Методы исследования

Рассмотрим серверное приложение, выполняющееся на высоком уровне приоритета при назначении алгоритма диспетчеризации RMMS-FP либо SMMS-FP. Будем полагать, что серверная задача относится к классу тяжелых объектов планирования и, соответственно, имеет приоритет выше, чем у любой легкой задачи. Клиентскими приложениями будем считать исключительно легкие задачи. Все тяжелые задачи в наборе полагаем независимыми. На рис. 1 представлено выполнение СРВ, удовлетворяющей всем перечисленным требованиям, причем исследуемая система состоит из $n = 4$ задач и имеет $m = 2$ процессора.

Поскольку τ_1 выполняется на высоком приоритете, обработке клиентского запроса может помешать лишь прерывание, либо событие принудительного вытеснения задачи сервера. Следует еще раз отметить, что используемый алгоритм планирования предусматривает

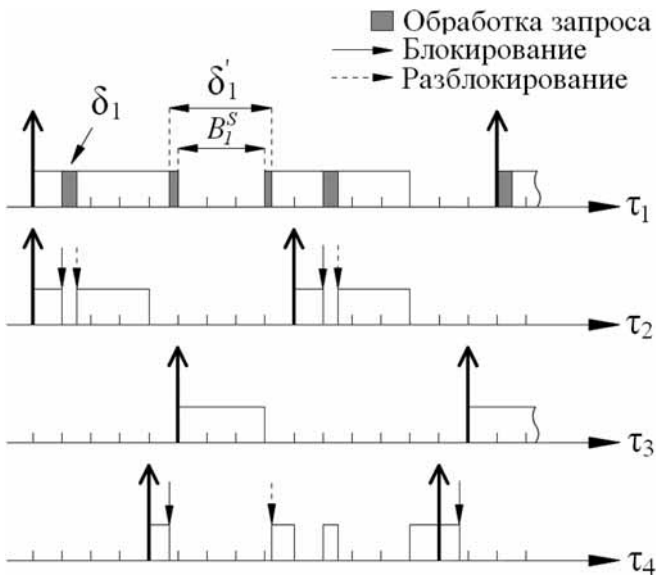


Рис. 1. Реализация сервера в виде высокоприоритетной тяжелой задачи при диспетчеризации методом RMMS-FP/SMMS-FP

применение принципа принудительных периодических вытеснений тяжелых задач. Степень влияния обработчиков прерываний на длительность обработки запроса сервером учитываться в настоящей статье не будет. Целью данной работы является исследование негативного влияния принудительных вытеснений на диспетчеризацию при наличии межзадачного взаимодействия.

Параметр δ_i (рис. 1) характеризует длительность обработки запроса серверной задачей τ_i, но не учитывает наличие принудительных периодических вытеснений. При оговоренных условиях δ_i может полагаться константой. Параметр $\delta'_i = \delta_i + B_i^S$ определяет максимальную длительность обработки запроса сервером. Если не учитывать переключения контекстов задач и влияние обработчиков прерываний (в конкретных случаях влияние этих параметров может считаться постоянным), параметр $\Delta_j \in [\delta_i;\ \delta'_i] = [\delta_i;\ \delta_i + \rho B_i^S]$ характеризует общую длительность блокирования клиентской задачи τ_j в ожидании завершения обработки запроса сервером. Параметр ρ характеризует число вытеснений, произошедших за время обработки запроса. В большинстве случаев $\rho = 1$, исключением является лишь выполнение условия $\delta_i > T_i^{BS} - B_i^S$, где T_i^{BS} — период принудительных вытеснений. Необходимо заметить, что именно принципиальное отличие параметров Δ_j и δ_i, определяемое наличием принудительных вытеснений серверной задачи на длительный интервал времени B_i^S, позволяет оценить возможность применения методов диспетчеризации RMMS-FP/SMMS-FP при наличии межзадачного взаимодействия. С этих позиций имеет

смысл именовать блокирование сервером клиентских задач на интервал времени δ'_i *длительной блокировкой*.

На рис. 1 представлен временной отрезок, характеризующийся отсутствием пропусков клиентскими задачами крайних сроков завершения. Соответственно, влияние длительных блокировок в данном случае будет отсутствовать. Рассмотрим сценарий, представленный на рис. 2, где серверной задачей является τ_1, а клиентскими — τ_2 и τ_4. Экземпляр задачи τ_2 порождается в тот момент, когда задача τ_4 претерпевает длительную блокировку на интервал времени $\Delta_4 = \delta'_1$. Задача τ_4, обладая наименьшим приоритетом, оказывается неспособной успешно завершиться до наступления крайнего срока. Очевидно, этого не могло бы произойти при использовании алгоритмов диспетчеризации RM-US/SM-US [4, 5], отличающихся лишь отсутствием принудительных периодических вытеснений тяжелых задач.

Отметим, что реализация сервера в качестве тяжелой задачи не соответствует наиболее распространенной практике, поскольку подразумевает совмещение опроса очереди клиентских запросов с выполнением протяженных во времени операций, не связанных непосредственно с обслуживанием клиентов. При проектировании СРВ разработчики зачастую стремятся выделить избыточную вычислительную нагрузку из серверной задачи в отдельные объекты диспетчеризации в целях уменьшения времени обработки клиентского запроса. Необходимо оценить возможность применения методов диспетчеризации RMMS-FP/SMMS-FP и степень влияния длительных блокировок на клиент-серверное взаимодействие для случаев, когда по каким-либо причинам подобное перераспределение вычислительной нагрузки не может быть реализовано. С этих позиций целесообразно воспользоваться методом оценки времени отклика, представленным в работе [7], несколько расширив его математический аппарат. Очевидно, изменения должны коснуться механизма учета длительных блокировок.

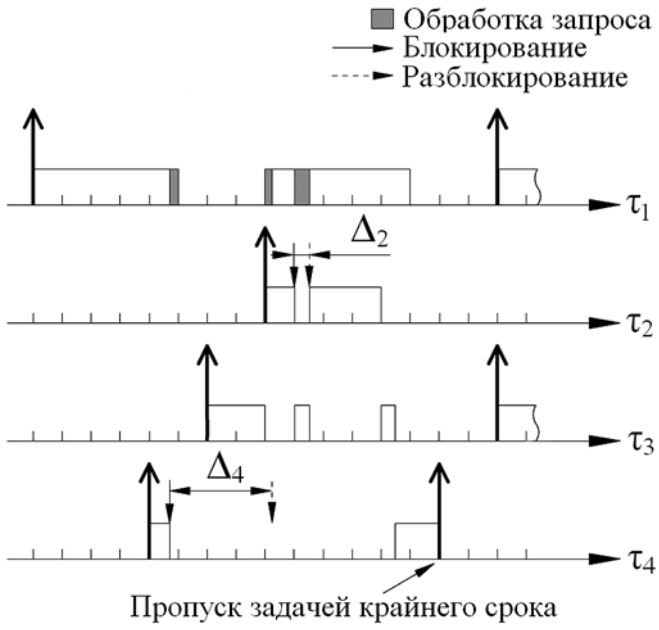


Рис. 2. Влияние длительных блокировок на отказы в диспетчеризации при реализации сервера в виде высокоприоритетной тяжелой задачи

Согласно основному положению теории анализа времени отклика, *система является успешно диспетчеризируемой, если время отклика R_i всех задач реального времени не превышает D_i.* Для алгоритмов глобальной статической диспетчеризации, к которым относятся методы RM-US/SM-US, справедлива оценка R_i, предложенная в работе [7]. Воспользуемся ее уточненным описанием [6]:

$$R_i = C_i + \left\lfloor \frac{\sum I_k^{NC} + \sum I_k^{DF}}{m} \right\rfloor, \qquad (1)$$

где $\sum I_k^{NC} = \sum\limits_{k \in hp(\tau_i)} I_k^{NC}(R_i, C_i)$ и $\sum I_k^{DF} = \sum\limits_{k \in \max(\tau_i, m-1)} I_k^{DF}(R_i, C_i)$, причем $\max(\tau_i, m-1) \subseteq hp(\tau_i)$ есть подмножество, состоящее не более чем из $m - 1$ задачи, у которых параметр $I_k^{DF}(R_i, C_i)$ максимален; $hp(\tau_i)$ — подмножество задач, чей приоритет выше, чем у τ_i. Максимальная интерференция (под *интерференцией* понимается влияние экземпляров различных задач на сроки выполнения), обусловленная высокоприоритетной задачей τ_k, определяется [7] следующим образом:

$$I_k^R(R_i, C_i) = \min(W_k^R(R_i), R_i - C_i + 1), \qquad (2)$$

где верхняя граница нагрузки на процессор, обусловленной выполнением задачи τ_k, есть $W_k^R(R_i) = N_k^R(R_i)C_k + M_k^R$, при $N_k^R(R_i) = \left\lfloor \frac{R_i + R_k - C_k}{T_k} \right\rfloor$ и $M_k^R = \min(C_k, R_i + R_k - C_k - N_k^R(R_i)T_k)$. Вклад в интерференцию $I_k^{NC}(R_i, C_i)$ экземпляров τ_k, начавших выполнение до момента порождения экземпляра τ_i, задан следующим образом [7]:

$$I_k^{NC}(R_i, C_i) = \min(W_k^{NC}(R_i), R_i - C_i + 1), \qquad (3)$$

где $W_k^{NC}(R_i) = N_k^{NC}(R_i)C_k + M_k^{NC}$ есть верхняя граница нагрузки на процессор подобных объектов диспетчеризации при $N_k^{NC}(R_i) = \lfloor R_i / T_k \rfloor$ и $M_k^{NC} = \min(C_k, R_i - N_k^{NC}(R_i)T_k)$. Разность обоих видов интерференции при учете (2) и (3) определяется выраже-

нием: $I_k^{DF}(R_i, C_i) = I_k^R(R_i, C_i) - I_k^{NC}(R_i, C_i)$. Рекуррентное выражение (1) может быть вычислено последовательно для всех τ_i при начальном приближении $R_i^0 = C_i$. Очевидно, анализ диспетчеризируемости СРВ выполняется после оценки R_i для всех задач реального времени.

Применение оценки (1) для алгоритмов диспетчеризации RMMS-FP/SMMS-FP становится возможным при замене всех тяжелых задач τ_i, использующих принцип принудительных вытеснений, моделями псевдозадач $\tau_{i,j}$, характеризующихся следующими параметрами:

$$C_{i,j} = T_i^{BS} - B_i^S,\ T_{i,j} = T_i^{BS}, \qquad (4)$$

где T_i^{BS} — период принудительных вытеснений. Интерференции обоих видов I_k^{NC} и I_k^{DF}, обусловленные выполнением псевдозадач $\tau_{i,j}$, вычисляются для низкоприоритетных задач в соответствии с выражениями (2) и (3) при учете (4). Очевидно, вычисление оценки времени отклика (1) для СРВ, выполняющихся с принудительными вытеснениями, остается возможным.

Принимая во внимание возможность длительных блокировок низкоприоритетных клиентских задач, имеет смысл учесть их в формуле (1):

$$R_i = C_i + \left\lfloor \frac{\sum I_k^{NC} + \sum I_k^{DF}}{m} \right\rfloor + K_q, \qquad (5)$$

где параметр K_q характеризует максимально возможный вклад серверной задачи τ_q в интерференцию клиентской задачи τ_i. Отметим, что появление дополнительного слагаемого в формуле (5) приводит к необходимости исследования выполнимости задач ПО СРВ при осуществлении наихудшего (*критического*) сценария. Учитывая возможность длительной обработки запросов сервером (охватывающей несколько интервалов принудительных вытеснений тяжелой задачи), имеем:

$$K_q = \delta_q + B_q^S \left\lceil \frac{\delta_q}{T_q^{BS} - B_q^S} \right\rceil, \qquad (6)$$

причем для критического сценария $K_q = \Delta_i$.

Применяя формулы (5) и (6) для низкоприоритетных клиентских задач и (1) — для всех остальных, оценим степень влияния длительных блокировок на диспетчеризацию мультипроцессорных СРВ исследуемыми методами.

Обсуждение результатов

На рис. 3 представлена диаграмма, отражающая влияние длительных блокировок на эффективность диспетчеризации задач в СРВ при $n = 10$ и $m = 4$. Параметр C характеризует общее число успешно выполняющихся наборов задач, соответствующих суммарной вычислительной нагрузке множества объектов диспетчеризации $U = \frac{\sum u_i}{m}$, где $u_i = C_i/T_i$ есть объем вычислительных ресурсов, потребляемых задачей. Размер выборки — 10 000 наборов для каждой рассматриваемой вычислительной нагрузки U. Под успешным выполнением периодических задач в системах жесткого реального времени понимается гарантированное предоставление необходимых вычислительных ресурсов в каждом периоде в объеме C_i до наступления крайнего срока их завершения. На рис. 3 обозначены: $j1$ — эффективность применения алгоритма диспетчеризации SM-US [5], $j2$ — эффективность применения алгоритма SMMS-FP, $j3$ и $j4$ аналогичны $j2$ при условии наличия длительных блокировок одной и двух легких клиентских задач соответственно (подразумевается существование единственного и общего тяжелого сервера в системе).

Очевидно, наличие тяжелого сервера негативно сказывается на диспетчеризации СРВ при использовании метода SMMS-FP. Уже при появлении единственной легкой задачи (рис. 3, $j3$), подверженной длительному блокированию, преимущество алгоритма SMMS-FP перед алгоритмом SM-US оказывается незначительным. Таким образом, применение подхода, основанного на принудительных вытеснениях, к тяжелой задаче, обладающей серверным функционалом, не рекомендуется.

Тем не менее степень негативного влияния длительных блокировок на эффективность диспетчеризации может быть снижена посредством увеличения

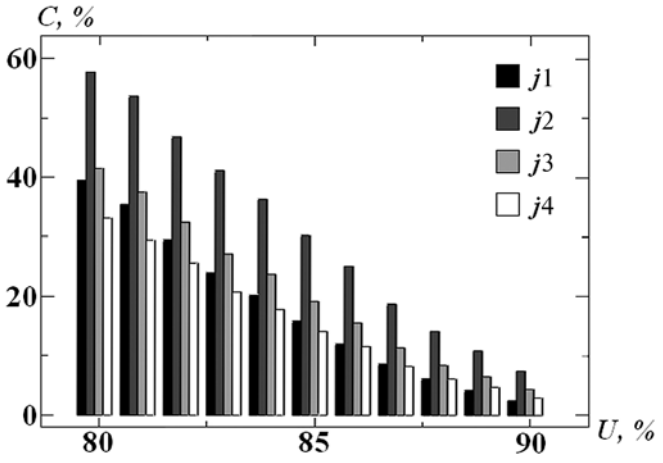


Рис. 3. Влияние длительных блокировок на диспетчеризацию легких задач при реализации сервера в виде тяжелой задачи

числа принудительных вытеснений сервера в отведенном ему периоде. Поскольку в рамках определенного периода тяжелая серверная задача потребляет неизменное количество вычислительных ресурсов C_i, увеличение числа принудительных вытеснений приведет к уменьшению их длительности и, соответственно снизит влияние на клиентские задачи. Справедливость данного утверждения подтверждает диаграмма, представленная на рис. 4 (число принудительных вытеснений в периоде равняется четырем, а *j*1—*j*4 соответствуют обозначениям, представленным на рис. 3). Необходимо отметить, что подобный шаг может оказаться неэффективным при учете переключений контекстов, а также при наличии большого числа клиентских задач. Рассмотрение данного вопроса является предметом будущих исследований.

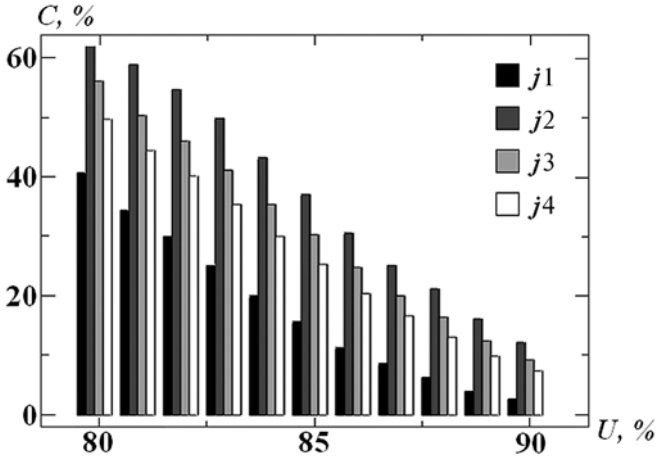


Рис. 4. Влияние длительных блокировок на диспетчеризацию легких задач при реализации сервера в виде тяжелой задачи с увеличенным числом принудительных вытеснений

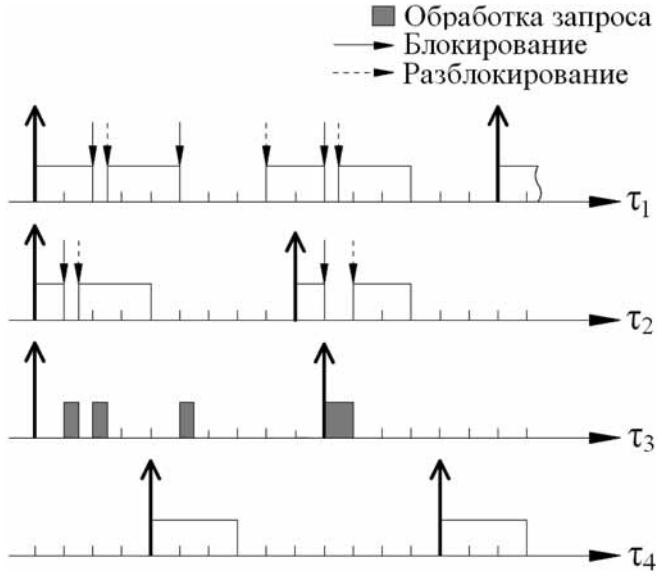


Рис. 5. Реализация сервера в виде легкой задачи

Реализация сервера в виде легкой задачи исключает возможность влияния описанного в работе эффекта. Ввиду высокой степени детерминированности процесса исполнения тяжелой задачи, влияние длительных блокировок на диспетчеризацию отсутствует даже при условии ее принадлежности к числу клиентских. Данное утверждение справедливо также при наличии нескольких клиентов легкого сервера, поскольку в данном случае применимы протоколы наследования и разделения приоритетов [8, 9]. На рис. 5 представлен пример реализации легкого сервера при условии наличия тяжелой клиентской задачи. Для рассматриваемого случая эффект от применения алгоритма диспетчеризации SMMS-FP соответствует серии экспериментов *j*2, представленной на рис. 3 и рис. 4. Очевидно, при подобной реализации межзадачного взаимодействия, исследуемые методы планирования обладают достаточной (максимальной с точки зрения предмета настоящего исследования) эффективностью для практического применения.

Детальный анализ полученных результатов позволяет сделать вывод о том, что использование методов RMMS-FP/SMMS-FP в мультипроцессорных СРВ при наличии межзадачного взаимодействия имеет особенности. В частности, использование принудительных периодических вытеснений имеет ощутимый потенциал при реализации сервера в виде легкой задачи. Следует также отметить, что при использовании в качестве сервера тяжелой задачи результаты оказываются существенно ниже. Тем не менее существуют серьезные основания полагать, что данный подход не имеет широкого распространения на практике. К тому же, как было показано выше, существует способ регулирования эффективности метода диспетчеризации при реализации тяжелого сервера.

Заключение

Представленные в статье результаты диспетчеризации задач в мультипроцессорных СРВ подтверждают предположение о применимости методов планирования с принудительными периодическими вытеснениями при наличии межзадачного взаимодействия. Однако использование данных методов оказывается затруднительным при реализации серверной задачи в форме тяжелого объекта диспетчеризации. Показано, что подобное явление неразрывно связано с возможностью возникновения длительных блокировок клиентских задач. В статье проведена оценка степени влияния указанного эффекта на диспетчеризацию в мультипроцессорных СРВ.

Рассмотрена возможность регулирования числа успешно выполняющихся наборов задач при реализации сервера в виде тяжелого объекта диспетчеризации посредством увеличения числа принудительных вытеснений в каждом периоде исполнения. Данная мера позволяет несколько снизить влияние длительных блокировок на эффективность диспетчеризации, но поскольку при исследовании не учитывались пере-

ключения контекстов, в будущем необходимо уточнение полученных результатов для этого случая.

Выяснилось, что негативное влияние длительных блокировок на эффективность использования алгоритмов диспетчеризации RMMS-FP/SMMS-FP в СРВ, реализующих легкие сервера, отсутствует. Их применение на практике в данных системах возможно в полном объеме, в соответствии с серией экспериментов *j*2, показанных на рис. 3 и 4. Использование исследуемых методов в реальных СРВ, в которых затраты на переключение контекстов отличны от нуля, в данной работе не рассматривались. Однако в первом приближении они могут быть соотнесены с серией экспериментов *j*2. Целью дальнейших исследований является применение методов снижения числа вытеснений легких задач, обусловленных принудительными вытеснениями, что может позволить несколько повысить эффективность диспетчеризации.

Список литературы

1. **Liu C. L., Layland J. W.** Scheduling algorithms for multiprogramming in a hard-real-time environment // Journal of the ACM. 1973. Vol. 20. № 1. P. 46—61.

2. **Douglass B. P.** Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns. Addison-Wesley, 1999. 750 p.

3. **Dhall S. K., Liu C. L.** On a Real-Time Scheduling Problem // Operating Research. 1978. Vol. 26. № 1. P. 127—140.

4. **Andersson B., Baruah S., Jonsson J.** Static-priority scheduling on multiprocessors // Proc. of the 22nd IEEE Real-Time Systems Symposium. Goteborg, Sweden. December, 2001. P. 193—202.

5. **Andersson B.** Global Static-Priority Preemptive Multiprocessor Scheduling with Utilization Bound 38% // Proc. of the 12th International Conference on Principles of Distributed Systems. Berlin. December, 2008. P. 73—88.

6. **Докучаев А. Н.** Задачи фи-типа как средство расширения границ применимости статической диспетчеризации в мультипроцессорных системах реального времени при высокой нагрузке // Системы управления и информационные технологии. 2011. № 3 (45). С. 224—229.

7. **Guan N., Stigge M., Yi W.** New Response Time Bounds for Fixed Priority Multiprocessor Scheduling // Proc. of 30th IEEE Real-Time Systems Symposium (RTSS 2009). Washington. December, 2009. P. 387—397.

8. **Borger M. W., Rajkumar R.** Implementing Priority Inheritance Algorithms in an Ada Runtime System. Technical Remailbox. CMU/SEI-89-TR-15, ESD-TR-89-23. Software Engineering Institute. Carnegie Mellon University. 1989. 54 p.

9. **Sha L., Rajkumar R., Lehoczky J. P.** Priority Inheritance Protocols: An Approach to Real-Time Synchronization // IEEE Transactions on Computers. 1990. Vol. 39. № 9. P. 1175—1185.

ИНФОРМАЦИЯ

ИТиС 2013

С 1 по 7 сентября 2013 г.

в Калининграде

состоится *37-я конференция-школа*

молодых ученых и специалистов

РОССИЙСКАЯ АКАДЕМИЯ НАУК

ИНСТИТУТ ПРОБЛЕМ ПЕРЕДАЧИ ИНФОРМАЦИИ им. А.А.Харкевича

"ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ — 2013" (ИТиС'13)

Конференция-школа ИТиС'13 будет своеобразной Universitas, состоящей из тематических семинаров по следующим основным научным направлениям:

- Seminarium mathematicum, или Математика и физика сложных систем;
- Seminarium explorationis datorum, или Структурные методы анализа данных и оптимизации;
- Seminarium protectionis informatiae, или Теория кодирования и ее приложения;
- Seminarium technologiarum retium, или Сетевые технологии и протоколы;
- Seminarium operis informationis in vivo, или Биология и биоинформатика;
- Seminarium linguisticum, или Компьютерная лингвистика.

Кроме seminaria, впервые на конференции-школе состоятся и учебные курсы (doctrinae) по "горячим" областям исследований.

Электронный адрес организационного комитета — itas@iitp.ru,

Сайт конференции — http://itas2013.iitp.ru/rss.xml

УДК 004.272

В. М. Пентковский, д-р техн. наук, проф.,
А. Ю. Дроздов, д-р техн. наук, проф.,
А. Д. Голенев, студент,
Ю. Н. Фонин, науч. сотр., e-mail: phonin@rambler.ru
С. Л. Шлыков, науч. сотр.,
Московский физико-технический институт (технический университет)

Ðàçðàáîòêà è ðåàëèçàöèÿ íà ÿçûêå Python ìåòîäèêè ïðåäñêàçàíèÿ îïòèìàëüíîé àðõèòåêòóðû ìèêðîïðîöåññîðà íà îñíîâå èñïîëüçîâàíèÿ íåéðîííûõ ñåòåé è ýìóëÿòîðà Graphite

Представлен метод предсказания оптимальной архитектуры микропроцессора с использованием нейронных сетей и эмулятора Graphite. Описан алгоритм предсказания, а также приведены аспекты его реализации на языке Python. Представлены результаты экспериментов по нахождению оптимальных архитектур для нескольких задач с указанием ошибки предсказания.

Ключевые слова: *архитектура, моделирование, нейронные сети, алгоритм предсказания*

Введение

Одним из основных инструментальных средств при проектировании микропроцессоров является программное моделирование. Определение необходимого уровня точности проектирования и моделируемых параметров микропроцессора является первоочередной задачей архитектора вычислительной системы.

На разных этапах проектирования применяются различные виды эмуляторов, реализующих компромисс между точностью моделирования и скоростью исполнения. К их числу относятся: потактовые эмуляторы, обладающие наивысшим уровнем точности и наименьшей скоростью исполнения; более быстрые функциональные эмуляторы; эмуляторы высокого уровня, моделирующие работу лишь узкого набора архитектурных элементов. Низкая скорость работы потактовых эмуляторов (замедление порядка 100 000—1 000 000 раз по сравнению с непосредственным исполнением программы на микропроцессоре) значительно ограничивает возможности их применения. На практике нередки ситуации, когда эксперимент длится около недели, а иногда и месяца. Дополнительно следует отметить, что на протяжении длительного времени сложность микропроцессорных систем интенсивно росла как за счет выполнения закона Мура, так и за счет эволюции архитектур вычислительных систем. Число параметров вычислительных систем, от которых зависит скорость исполнения программ, быстро растет, что в большинстве случаев ограничивает область применения потактовых эмуляторов только отдельными небольшими фрагментами программ. Дополнительные сложности вносит в моделирование микропроцессоров недавно установившийся тренд, согласно которому разработчики стремятся интегрировать все больше процессорных ядер на одном кристалле. Предпочтение при этом отдается более простым ядрам, с отсутствием динамического планирования, с упрощенными механизмами предсказания переходов и т. д. (так называемые *many-core* процессоры). Эти слож-

ности связаны как с отсутствием параллельных эмуляторов промышленного качества, так и с резко увеличившейся сложностью моделируемых систем. В современных архитектурах появляются новые параметры, такие как тип протокола поддержки когерентности кэшей, тип коммуникационной сети на кристалле и др. В качестве наиболее известных примеров подобных архитектур можно привести процессоры Intel MIC и модельную линию VLIW процессоров фирмы Tilera. Несмотря на то что в настоящее время выпускают в основном симметричные многоядерные процессоры, заметна тенденция к созданию гетерогенных систем. Такие системы содержат как процессорные ядра различных типов (RISC, DSP, VLIW, SIMD), так и специальные аппаратные ускорители.

Целью архитектора микропроцессорных систем является установление характера взаимодействия между параметрами моделируемой архитектуры и изучение их влияния на производительность вычислительной системы с тем, чтобы на основе полученной информации принять решение об используемой архитектуре. Размер множества всех доступных конфигураций вычислительной системы экспоненциально зависит от размерности пространства параметров, что делает неосуществимым на практике исчерпывающий поиск среди всех возможных комбинаций. Обычно выбирается достаточно узкое подмножество, являющееся подпространством пространства конфигураций. При этом подходе моделируются только те параметры, которые оказывают наибольшее влияние на производительность системы. Выбор этих параметров целиком зависит от опыта архитектора. Однако возрастающая сложность современных микропроцессоров не всегда позволяет интуитивно выбрать такое подмножество параметров, что изменение только данных параметров приведет к нахождению оптимальной архитектуры. Известны случаи, когда учет ранее игнорируемого параметра приводил к диаметрально противоположным решениям. Кроме того, появляются новые аппаратные решения (например, транзакционная память, оптические коммуникационные сети, интегрированные на кристалле и др.), использование которых позволяет добиться существенного повышения производительности.

В свете изложенного выше возникает необходимость в алгоритмах и программных системах для исследования пространства конфигураций вычислительной системы в полуавтоматическом (автоматизированном) режиме. Такие системы, с одной стороны, должны автоматизировать работу по конфигурированию архитектур, компиляции, запуску и анализу результатов. С другой стороны, они должны предоставить архитектору возможность задать ограничения, определить набор варьируемых параметров, выбрать стратегии поиска и необходимый уровень точности, а также проанализировать промежуточные результаты моделирования и, при необходимости, остановить поиск. При этом важно отметить два аспекта процесса: во-первых, необходимо максимально автоматизировать все остальные этапы процесса исследования, не связанные с формулировкой ограничений накладываемых на решение; во-вторых, скорость процесса поиска решения не должна ограничивать размер исследуемого множества конфигураций при заданном уровне точности.

В настоящей публикации сформулированы требования, предъявляемые к методологии полуавтоматического поиска в пространстве конфигураций вычислительной системы, предложен вариант реализации и описан опыт ее применения на практике.

Обзор программных эмуляторов Graphite и Sniper

Система Graphite [1] представляет собой программный эмулятор многоядерных архитектур уровня приложений. Эмуляция заключается в исполнении многопоточной программы и моделировании ее работы на многоядерной архитектуре, определенной моделями эмулятора и параметрами времени исполнения. Эмулятор Graphite привязывает каждый поток приложения к ядру эмулируемой архитектуры. Каждое моделируемое ядро может исполнять одновременно только один поток. Таким образом, число одновременно запущенных на исполнение потоков не может превысить число ядер, заданное в конфигурации целевой архитектуры. На одном или нескольких узлах кластера запускается несколько процессов (каждый поток исполняется на одном из процессов). Синхронизация и обмен данными осуществляются при помощи протокола TCP/IP. Потоки приложения являются частью этих процессов и управляются ОС хост-машины.

Система Graphite имеет модульную архитектуру, в которой каждый компонент реализован как заменяемый модуль, который имеет хорошо определенный интерфейс для взаимодействия с другими модулями в системе.

Каждый модуль конфигурируется через набор параметров, значения которых задаются в конфигурационных файлах. Система использует динамическую бинарную трансляцию чтобы делать вставки обратных вызовов эмулятора в код приложения. В частности, Graphite переписывает код, оперирующий ссылками на память и системными вызовами, а также генерирует поток инструкций для моделирования. Множество инструкций, например, арифметические и логические операции, не нужно эмулировать, их можно исполнять непосредственно на хост-машинах, получая существенное ускорение. Graphite использует Pin [2] в качестве фронт-энда, однако в силу модульной архитектуры может быть использован и другой инструментарий динамической трансляции. Компоненты, предоставляемые бэк-эндом Graphite, можно разделить на функциональные и средства для моделирования.

Модель ядра используется для моделирования процессорного конвейера, модель памяти — для иерархии памяти. Модель сети обрабатывает маршрутизацию сообщений через сеть на кристалле и учитывает разнообразные задержки. Важно отметить, что модели взаимодействуют друг с другом, чтобы определять

стоимость событий в приложении. Например, модель памяти использует модель сети чтобы определить продолжительность операций с памятью, а модель ядра определяет время исполнения арифметических и логических операций, используя задержки, значение которых вычисляется в модели памяти. Ключевой особенностью эмулятора является нестрогая схема синхронизации. Каждый поток эмуляции имеет собственные локальные часы и синхронизируется с остальными только во время наступления специальных событий. Одним из основных достоинств Graphite является способность исполнять многопоточное приложение без модификаций на нескольких узлах кластера. Для этого Graphite обеспечивает:

- единое адресное пространство;
- интерфейс с ОС;
- интерфейс с библиотекой потоков;
- передачу сообщений.

Эмулятор Sniper [3], непосредственно используемый в данной работе, представляет собой быстрый параллельный эмулятор многоядерных микропроцессоров, реализующих архитектуру x86. Он основан на применении интервальной модели ядра [4] и инфраструктуре Graphite.

Sniper является более надежным и стабильным, чем Graphite. Он лишен также других недостатков, свойственных Graphite: в частности, поддерживаются динамические библиотеки, и нет необходимости в перекомпиляции исходных кодов приложения (можно использовать бинарный файл без специальной компиляции для системы Sniper). Ключевой особенностью данного эмулятора является интервальная модель, повышающая уровень абстракции в архитектурной эмуляции.

Описание особенностей алгоритма автоматического поиска оптимальной архитектуры

Практически в любой инженерной дисциплине при проектировании преобладает подход "сверху вниз". Одним из самых ярких примеров является разработка программного обеспечения, характеризуемая постепенным понижением уровня абстракции от высокоуровневого описания архитектуры продукта до реализации конкретного функционала программы. При построении предлагаемой в настоящей работе методологии хотелось бы придерживаться подобного подхода, поскольку это позволило бы на конкретных этапах проектирования вычислительной системы абстрагироваться от более низкоуровневых деталей.

В процессе длительных исследований в мире создано широкое портфолио различных технологий и средств эмуляции, начиная от специальных систем, использующих ПЛИС, заканчивая быстрыми execution-driven эмуляторами. Данные технологии заполняют спектр соотношений время/точность и имеют различные накладные расходы в применении. Например, может потребоваться выделенная под эмуляцию ферма микропроцессоров или специальная система ПЛИС. Может появиться необходимость в использовании специализированного высокоуровневого языка (например, [5]) или языка описания аппаратуры, такого как VHDL или Verilog. Это, в свою очередь, может привести к значительному увеличению расходов на моделирование.

Наиболее эффективным является итеративный процесс исследования с возможностью возврата к предыдущим итерациям. При таком подходе одна и та же методология применяется последовательно с использованием различных эмуляторов, обладающих различным соотношением скорости и точности. При этом на каждой новой итерации происходит понижение уровня абстракции и включение в рассмотрение новых деталей архитектуры.

Резюмируя изложенное ранее, отметим, что методология должна быть нейтральной по отношению к используемой технологии эмуляции. Более того, следует подчеркнуть необходимость использования различных эмуляторов. Приведем пример. Технология статистической эмуляции [6] является относительно новой и обладает экстремальным соотношением точность/скорость. В процессе статистической эмуляции наблюдается поведение системы на некоторых участках кода, и, согласно полученным данным, конечный результат поведения всей программы получается при помощи экстраполяции. Для оценки эффективности работы вычислительной системы в первом приближении можно использовать данную технологию. После этого можно отсеять решения, не удовлетворяющие заданным ограничениям и сделать множество поиска более узким. Следующими далее этапами будут применения более медленных функциональных эмуляторов, не использующих статистические техники. Конечный этап — подробное моделирование при помощи потактового эмулятора. Подобный подход позволяет не только ускорить процесс исследования, но и упростить работу архитектора, представляя возможность гибко контролировать уровень точности и брать в расчет только важные для конкретного этапа проектирования детали.

Несмотря на то что подход, описанный выше, значительно сокращает время эмуляции, вопрос большой длительности моделирования каждой отдельной конфигурации остается открытым. На каждом этапе необходимо провести большое число эмуляций, если целью является всеобъемлющий поиск. Появляется необходимость в технологии, альтернативной описанной выше, позволяющей избежать большого числа отдельных эмуляций. В качестве возможных решений предлагаются различные варианты эвристического поиска и предсказательное моделирование. В данной работе дается адекватная оценка применимости последнего. Основным недостатком эвристического поиска является отсутствие возможности охватить все пространство конфигураций вычислительной системы, поскольку он не уменьшает число эмуляций принципиальным образом, в отличие от предсказа-

тельного моделирования. Технологии предсказательного моделирования основаны на математической статистике и методах машинного обучения. Применение данной технологии в методологии поиска оптимальных конфигураций вычислительных систем заключается в следующем: рассматривается эмулятор как черный ящик, на вход которому подают параметры конфигурации; на выходе при этом получаем результаты эмуляции, представленные в виде набора выходных параметров, таких как, например, время работы в тактах, число промахов при обращении к кэш или число выполненных условных переходов.

Цель предсказательного моделирования — построить на основе имеющихся пар векторов входных параметров и точных значений выходных метрик, полученных в результате эмуляции, математическую модель вычислительной системы, представленной в виде черного ящика (т. е. выполнить регрессию), что позволяет в дальнейшем аппроксимировать значения выходных метрик, взамен проведения дорогостоящих эмуляций. Регрессионные методы достаточно хорошо изучены и применяются в различных сферах человеческой деятельности. Существует целый ряд регрессионных моделей, в числе которых:

- линейная регрессия;
- аппроксимация сплайнами;
- радиальные базисные функции;
- искусственные нейронные сети.

Основываясь на предположении, что функция "черного ящика", характеризующая модель вычислительной системы, имеет существенную нелинейность, что демонстрируют проведенные научным сообществом исследования, не рекомендуется использовать линейную регрессию в качестве предсказательного механизма. Кроме того, против аппроксимации полиномами играет тот факт, что ими можно аппроксимировать значения неизвестной функции с высокой точностью только для входных значений, лежащих в пределах обучающей выборки. Вне ее пределов интерполяция полиномами имеет низкую точность. Идеальным выбором являются искусственные нейронные сети [7], которые хорошо изучены и нашли широкое применение в науке и технике. Кроме того, нейронные сети не требуют никаких предварительных знаний о природе аппроксимируемой функции и хорошо приближают любые нелинейные функции. Строго доказано [7], что произвольная функция может быть приближена трехслойной нейронной сетью со сколь угодно высоким уровнем точности. Для повышения точности предсказаний значения входных параметров и выходных метрик необходимо преобразовать специальным образом перед обучением нейронной сети или перед подачей параметров на входы сети во время процесса предсказательного моделирования. Кроме того, существуют специальные методы, также направленные на повышение точности нейронных сетей. Некоторые из них применялись при реализации методологии на практике и описаны в следующем разделе.

Для обучения предсказательного механизма необходима обучающая выборка, состоящая из точных результатов эмуляций. Качество этой выборки непосредственно влияет на точность получаемых в процессе обучения моделей. Множество начальных точек, которые необходимо подвергнуть моделированию, можно генерировать каждый раз случайным образом, однако можно использовать и адаптивные методы. Например, итеративный подход, при применении которого на каждой новой итерации точки из пространства конфигураций выбираются с использованием информации о предыдущих результатах моделирования. Данный процесс можно осуществлять одновременно с уменьшением области пространства конфигураций, в которой проводится поиск. Отметим, что данная методология в первую очередь ориентирована на снижение временных затрат на моделирование и повышение продуктивности работы инженеров, что, в свою очередь, получается в результате приношения в жертву точности. При грамотном выборе метода регрессии и генерации начальных точек в качестве обучающей выборки можно выбирать подмножество конфигураций, составляющее малую долю от всего множества конфигураций.

Использование регрессионных моделей кроме преимущества в скорости дает пользователю рассматриваемой методологии еще одно преимущество. Оно заключается в охвате всего пространства конфигураций, что делает возможным выполнение некоторых вспомогательных задач. Иногда бывает полезно осуществить кластеризацию, т. е. сформировать из конфигураций, схожих по значениям выходных параметров, отдельные группы и проводить более детальное моделирование конфигураций, принадлежащих интересующей исследователя группе, например, с использованием более медленного и более точного эмулятора. Внутри этой выделенной группы можно проводить процедуры оптимизации по данным параметрам, а также более детальный анализ изменения входных параметров в рамках группы.

Следует отметить, что любые регрессионные методы обладают некоторым недостатком. Самая лучшая по какой-либо характеристике конфигурация может иметь значение соответствующей метрики, значительно отличающееся от значений большинства конфигураций. Поскольку, как правило, обучающие выборки захватывают закономерности, характерные для большого числа конфигурации, невозможно предсказать значения метрик для конфигурации, сильно отличающейся от остальных (например, представляющей собой глобальный максимум значения какой-либо метрики), с высокой точностью. Такая конфигурация может иметь гораздо большую производительность, чем конфигурации, использованные для построения модели. Данная методология не подходит для поиска единственной оптимальной конфигурации. Ее назначение заключается в выделении особенностей, характерных для большого числа конфигураций, определении трендов, построении Парето-фронтов, иллюстрирующих воз-

можности нахождения компромисса между конфликтующими задачами, стоящими перед архитектором.

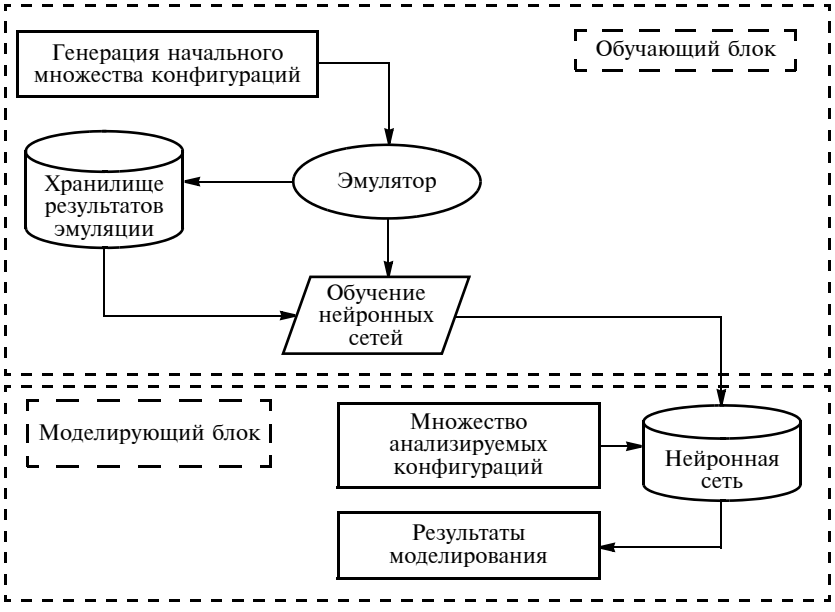
Дополнительно отметим, что данная методология может быть интересна не только для инженеров, проектирующих вычислительные системы, но и для разработчиков программного обеспечения. Например, в качестве входных параметров можно использовать наличие или отсутствие ключей компилятора, включение/исключение каких-либо компонентов из программного комплекса или подбор алгоритмов, реализующих альтернативные стратегии решения одной задачи. Отметим, что эта методология может быть полезна программистам, разрабатывающим программное обеспечение для еще не реализованной в кремнии архитектуры, разработчикам суперкомпьютерных систем (так называемая одновременная разработка аппаратного и программного обеспечения *hardware-software codesign*). Представленная методология является гибкой по отношению к целям и задачам разработчика системы и используемым инструментам. Функциональная схема методологии в общем виде представлена на рисунке.

Немаловажными аспектами, характеризующими эффективность методологии, являются стоимость ее применения и сложность практической реализации. С этих позиций представленная выше методология имеет большое преимущество, поскольку она не предполагает необходимости использования какого-либо проприетарного программного обеспечения. В процессе длительных исследований, проводимых в университетах и коммерческих организациях был создан большой объем программного обеспечения с открытым исходным кодом (в том числе с лицензиями, позволяющими использование в коммерческих целях) для эмуляции микропроцессорных систем и вычислительных кластеров, включающего в себя полный спектр технологий от потактовых эмуляторов до быстрых execution-driven эмуляторов. Кроме того, разработано достаточно много инструментальных средств и библиотек с открытым исходным кодом для статистического анализа данных, построения регрессий, машинного обучения. В процессе работы с применением библиотек и утилит с открытым исходным кодом с участием авторов было создано подобное рабочее окружение, реализующее идеи данной методологии. В качестве языка программирования был использован Python.

Аспекты реализации системы

Рабочее окружение системы, реализующее принципы представленной выше методологии, построено путем связывания утилит с открытым исходным кодом с помощью скриптового языка (в данном случае Python). Такой способ построения дает преимущество прозрачной интеграции дополнительных эмуляторов и утилит в цепочку инструментальных средств (далее для краткости — инструментов) или исключения существующих из нее. Очевидно, что применение проприетарного программного обеспечения в рабочем окружении также возможно. Однако использование только открытых программных продуктов в такой системе уменьшает накладные расходы на ее развертывание.

Для упрощения интеграции каждого нового инструмента необходимо использовать единый формат данных для передачи по цепочке инструментов. В общем случае для добавления новой утилиты необходимо реализовать в связующей логике функциональные возможности для преобразования данных из унифицированного для рабочего окружения формата в формат, используемый конкретным инструментом. В качестве выходных параметров, определяющих эффективность работы системы, были выбраны: общее число тактов; рассеиваемая мощность; площадь кристалла. В предлагаемом варианте рабочего окружения использовался единственный эмулятор, в качестве которого был выбран Sniper, описанный выше (поскольку одной из целей работы была оценка его возможностей). Число варьируемых параметров и точность моделей, доступных в Sniper, достаточны для проведения предварительных архитектурных исследований. Они позволяют весьма точно отразить поведение микропроцессора на высоком уровне. Из выходных данных, непосредственно полученных эмулятором, можно извлечь число тактов. Для расчета площади кристалла и рассеивае-



Функциональная схема моделирующего комплекса, использующего нейронные сети

мой им мощности, выходные данные подавались на вход утилиты McPAT [8]. Введение дополнительных варьируемых параметров и выходных метрик сопровождается минимальными изменениями в коде, поскольку формат данных, которым обмениваются компоненты друг с другом, тривиален и описан централизованно. Процесс расчета рассеиваемой мощности и площади кристалла является опциональным, поскольку, как показала практика, несколько замедляет общий процесс эмуляции. По этой причине в управляющем скрипте предусмотрена опция отключения моделирования мощности и площади.

Для хранения результатов эмуляции и их пост-обработки желательно использовать формат, являющийся устоявшимся стандартом хранения данных. Данные, хранящиеся в таком формате, удобно систематизировать, выбирать и обрабатывать стандартными процедурами. В данном случае было принято решение использовать легковесную базу данных SQLite3, для работы с которой в Python существует удобный для использования модуль. Значения параметров архитектуры, соответствующие им значения целевых параметров, полученных непосредственно в результате эмуляции или дополнительной обработки результатов, упаковываются в вектора, образующие обучающую выборку, используемую для построения регрессионной модели. Для быстрого прототипирования предсказательного механизма было принято решение использовать готовый фреймворк для машинного обучения с открытым кодом. В данной работе использовался PyBrain [9]. Он написан на языке Python, что в нашем случае упрощает его интеграцию в рабочее окружение. Скриптовая реализация является его основным минусом, несмотря на то, что встроенные в нее алгоритмы имеют хорошую для интерпретируемого языка производительность, поскольку работает намного медленнее, чем выполненная с помощью компилируемого языка. Однако в PyBrain существует возможность замены медленной, скриптовой реализации более быстрой, написанной на C++, с сохранением остального интерфейса с Python. Отметим, что данная реализация не так стабильна как написанная на Python и поддерживается одним человеком, поэтому использовать ее следует с осторожностью. Дополнительно модульная архитектура PyBrain позволяет гибко настраивать существующие и создавать собственные алгоритмы и компоненты для машинного обучения, пользуясь готовыми функциями как строительными блоками.

Обучение нейронной сети заключается в подстройке весовых коэффициентов. Очевидно, что нейронная сеть должна давать верные предсказания для результатов, на которых она не обучалась. По этой причине для обучения и валидации нейронной сети используются различные наборы данных.

Ключевыми параметрами нейронной сети являются число скрытых слоев, число узлов в скрытом слое, моментум и темп обучения (шаг градиента).

В предлагаемой реализации используется трехслойная нейронная сеть, имеющая топологию "каждый с каждым" и единственный сигмоидный скрытый слой. Число узлов в скрытом слое равно 16, шаг градиента — 0,001, моментум — 0,5. Известно, что аппроксимация полиномами дает очень точные результаты в пределах обучающей выборки, но плохо интерполирует за ее пределами. Похожее затруднение существует и у нейронных сетей. Для его предотвращения используется кросс-валидация и ранняя остановка. Обычно точность предсказаний получается более высокой в том случае, когда не достигается минимальная ошибка на обучающей выборке. Для этого разбиваем все данные на две части разного размера: обучающую выборку (бóльшая часть) и набор данных для валидации. Фиксированное число итераций настройки весов назовем эпохой. После каждой эпохи тестируем нейронную сеть и, если квадрат ошибки не уменьшается, прекращаем обучение. Может возникнуть ситуация, когда в течение нескольких эпох квадрат ошибки не уменьшается, а потом снова начинает уменьшаться. По этой причине процесс обучения контролируется значением специальной переменной, передаваемой через командную строку и показывающей, в течение скольких эпох допустимо ждать улучшения результатов валидации. Полученная нейронная сеть с наименьшей ошибкой далее используется для предсказания. Однако таким образом уменьшается обучающая выборка, и, как следствие, снижается и точность получаемых моделей. Для предотвращения снижения точности используется кросс-валидация. Вся обучающая выборка разбивается на N равных частей, после этого создается N различных нейронных сетей. Сгруппируем их вместе и назовем ансамблем. Далее каждой сети присваивается один из N наборов таким образом, что каждая из N нейронных сетей обладает уникальным набором, который будет служить для нее выборкой для тестирования. Оставшиеся $(N-1)$ наборов образуют для каждой сети обучающую выборку. Далее обучаем нейронные сети на соответствующих выборках. Таким образом, совокупно сети обучаются и валидируются сразу на всей доступной выборке. В случае же единственной сети для обучения и валидации используются разные входные данные. Для предсказания входной вектор подается на вход каждой нейронной сети в ансамбле, в качестве результата выступает среднее арифметическое выходное значение по ансамблю. Такая практика широко применяется в машинном обучении. Архитектурные параметры преобразуются перед подачей на вход нейронных сетей. Область определения каждой компоненты вектора входных данных отображается на отрезок [0, 1] при помощи минимаксной нормировки, которая проводится по формуле:

$$A_{\text{norm}} = \frac{(A - A_{\min})}{(A_{\max} - A_{\min})},$$

где A — начальное (не нормализованное) значение параметра; $A_{\max}$ — максимально допустимое значение параметра; $A_{\min}$ — минимально допустимое значение параметра; A_{norm} — нормализованное значение пара-

метра. Аналогичные манипуляции совершаем и по отношению к выходным значениям.

Следует отметить, что все нейронные сети имеют один выход, т. е. при одновременной работе с разными контролируемыми параметрами, например, энергопотреблением и числом циклов, нейронные сети создаются отдельно для каждого из них. Такой подход позволяет повысить точность предсказаний. Для ускорения работы, при наличии достаточного количества вычислительных ресурсов, можно использовать параллелизм уровня задач, представленный в процессе обучения нейронных сетей. Построение независимых регрессионных моделей можно проводить параллельно. При этом каждая такая модель вычислений хорошо отображается как на многоядерные (многопроцессорные) системы с общей памятью, так и на распределенные системы, такие как сети рабочих станций, вычислительные кластеры, облачные системы. Таким образом, для каждой нейронной сети создается объект, который содержит статус выполнения процедуры обучения. Результат выполнения процедуры возвращается асинхронно и вызывающая сторона может получить его, периодически опрашивая данный объект на предмет завершения операции. Такая модель упрощает параллельную обработку данных.

Для реализации многопараметрической оптимизации была использована библиотека DEAP [10] (*Distributed Evolutionary Algorithms for Python*). Основные ее плюсы — эффективная работа на многоядерных узлах и кластерах, а также возможность гибкой настройки параметров эволюционных стратегий.

По результатам работы алгоритмов на выходе имеем либо оптимальную конфигурацию для поиска по одному параметру, либо Парето-множество для многокомпонентной оптимизации. При этом создается исполняемый модуль, содержащий словарь массивов входных параметров и соответствующих им результатов. Его можно динамически импортировать в другие модули для дальнейшей обработки, например, для визуализации.

Моделирование

Для оценки точности описываемого в данной статье метода было проведено моделирование работы четырех приложений из бенчмарка Splash2, который является стандартным многопоточным бенчмарком для исследования многопроцессорных систем с централизованной или распределенной общей памятью. Использованная версия бенчмарка написана с помощью библиотеки POSIX Threads. Для моделирования использовались следующие приложения: FFT, FMM, Barnes, Cholesky. В процессе моделирования эмулировалась работа данных приложений на наборе многоядерных микропроцессорных архитектур на основе x86-совместимых процессоров. Архитектуры отличались друг от друга следующими параметрами:

- число ядер;
- размеры кэш-памяти каждого из трех уровней иерархии кэш;
- ассоциативность кэш (отдельно по каждому уровню);
- размеры блоков кэш-памяти.

Для каждого параметра был задан конечный диапазон допустимых значений. Моделирование проводилось на полном наборе значений параметров. Результаты моделирования в объеме 10 % использовались в качестве обучающей выборки для нейронной сети. Остальные 90 % использовались в качестве эталона для оценки погрешности предсказания нейронной сети.

Особенности процесса моделирования. Основной трудностью для выполнения целей моделирования являлась необходимость моделировать все конфигурации. Это необходимо для того, чтобы в дальнейшем была возможность оценить точность предсказаний результатов моделирования, выдаваемых нейронными сетями, общее число которых составляет 2160. По этой причине было принято решение использовать минимально допустимые объемы входных данных для приложений, при которых не происходит искажение производительности. При таком подходе, например, функция чтения памяти или вывода результатов не становится самой медленной. Число варьируемых параметров было выбрано небольшим, для того, чтобы была возможность провести эмуляцию всех конфигураций в разумный срок, учитывая имеющиеся в наличии вычислительные средства.

Результаты моделирования и автоматического предсказания оптимальной архитектуры. Моделирование с использованием нейронных сетей обладает явным преимуществом по скорости по сравнению с традиционным моделированием. Полная эмуляция всех конфигураций для каждого из приложений заняла около пяти суток. На получение обучающей выборки, составляющей 10% от всего объема конфигураций, было потрачено около 12 часов. Время обучения нейронных сетей зависит от желаемого уровня точности. Однако есть вероятность, что начиная с определенной итерации обучения уровень точности перестанет улучшаться.

Точность технологии моделирования с использованием нейронных сетей проиллюстрирована вычислением средней относительной ошибки. Ошибка предсказания результатов моделирования нейронными сетями для отдельной конфигурации E_i рассчитывалась по следующей формуле:

$$E_i = \frac{|P_i - S_i|}{S_i},$$

где S_i и P_i — результаты эмуляции и предсказания результатов эмуляции нейронными сетями для i-ой конфигурации соответственно. Для каждого приложения

вычисляется средняя относительная ошибка по следующей формуле:

$$\langle E \rangle = \frac{\sum_{i=1}^{N_{config}} E_i}{N_{config}},$$

где N_{config} — общее число конфигураций.

Результаты вычислений средних относительных ошибок представлены ниже.

Приложение	Ошибка, %
Splash2.barnes	10,6
Splash2.cholesky	14,0
Splash2.fft	5,9
Splash2.ffm	8,1
Средняя по бенчмарку ошибка предсказания	9,65

Заключение

В работе описан метод предсказания оптимальной архитектуры многопроцессорных систем с использованием нейронных сетей и эмулятора Graphite. В результате проведенных экспериментов продемонстрировано, что за счет использования нейронных сетей можно на порядок сократить суммарное время моделирования, необходимое для определения оптимальной архитектуры. При этом средняя ошибка предсказания не превышает 9,65 %, что является вполне приемлемым результатом, так как программы, для которых актуально использование многоядерных систем, ускоряются на 100 % и более.

Данная работа выполнена в рамках гранта, выделенного в соответствии с постановлением Правительства России № 220 от 09.04.2010 г.

Список литературы

1. **Miller J. E., Kasture H., Kurian G. et al.** Graphite: A distributed parallel simulator for multicores // The 16th IEEE International Symposium on High-Performance Computer (HPCA'10). Bangalore, India, January 2010. P. 1—12.

2. **Reddy V., Settle A., Connors D. A., Cohn R. S.** Pin: a binary instrumentation tool for computer architecture research and education. URL:http://www4.ncsu.edu/~efg/wcae/2004/submissions/connors.pdf

3. **Carlson T. E., Heirman W., Eeckhout L.** Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations URL: http://www.exascience.com/wp-content/uploads/2011/09/Sc2011carlson-final.pdf

4. **Genbrugge D., Eyerman S., Eeckhout L.** Interval simulation: Raising the level of abstraction in architectural simulation // Proc. of the 16th IEEE International Symposium on High-Performance Computer Architecture (HPCA). 2010. P. 307—318.

5. **Nikhi R. S.** What is bluespec? // ACM SIGDA Newslette 2009. Vol. 39. № 1. P. 1—1. URL: http://dl.acm.org/citation.cfm?id=1862868

6. **Hamerly G., Perelman E., Calder B.** How to use simpoint to pick simulation points // SIGMETRICS Performance Evaluation Review. 2004. Vol. 31. № 4. P. 25—30.

7. **Mitchell T.** Machine Learning. New York: McGraw-Hill, 1997. URL: http://www.bibsonomy.org/bibtex/23e79734ee1a6e49aee02ffd108224d1c/flint63

8. **Li S., Ahn J. H., Stronget R. D.** Mcpat: an integrated power, area, and timing modeling framework for multi-core and many-core architectures // Proc. of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO 42. New York, NY, USA: ACM, 2009. P. 469—480.

9. **Schaul T., Bayer J., Wierstra D.** PyBrain // Journal of Machine Learning Research. 2010. Vol. 11. P. 743—746.

10. **Rainville F.-M. D., Fortin F.-A., Gardner M.-A.** Deap: A python framework for evolutionary algorithms URL: http://vision.gel.ulaval.ca/~cgagne/pubs/deap-gecco-2012.pdf

ИНФОРМАЦИЯ

УДК 004.75

С. К. Шикота, программист, **А. Ю. Меньшутин,** канд. физ.-мат. наук, мл. науч. сотр.,
Л. Н. Щур, д-р физ.-мат. наук, зав. отд., Научный центр РАН в Черноголовке,
e-mail: shchur@chg.ru

Ïðîãðàììíî-àïïàðàòíàÿ ïëàòôîðìà äëÿ íàó÷íûõ èññëåäîâàíèé â îáëàñòè âû÷èñëèòåëüíîé ôèçèêè

Обсуждаются пути создания программно-аппаратной платформы, призванной облегчить пользователю-предметнику доступ к сложным вычислительным ресурсам и их использование. Простой и интуитивно понятный интерфейс к вычислительным ресурсам предоставляет пользователю возможность сосредоточиться исключительно на решении задачи в предметной области.

Ключевые слова: *совместная работа, веб 2.0, облачные вычисления, визуализация*

Введение

Современные исследования в области вычислительной физики базируются на вычислительных экспериментах и численном моделировании. Проведение передовых поисковых научных исследований налагает большие требования как на ресурсы машинного времени, так и на ресурсы оперативной памяти, а также на объем хранимой информации и на полосу обмена информацией между устройствами. Необходимость в вычислительных ресурсах высокой производительности, а также в больших объемах хранимой информации, указывают на целесообразность использования распределенных программно-технических средств, таких, например, как грид [1] и вычислительные облака [2]. Современный подход к созданию эффективных средств доступа к таким ресурсам, средств мониторинга и контроля процесса выполнения исследований состоит в разработке специализированных порталов и программно-аппаратных платформ.

В работе [3] подчеркнуто, чтобы усилия по исследованию и развитию информационных систем были наиболее успешны, необходимо понять, почему некоторые проекты создания научных и инженерных платформ способны изменить стиль выполнения фундаментальных научных исследований в определенном сообществе людей. Авторы выделили четыре основных аспекта устойчивости развития таких платформ. Первый аспект — это финансирование проекта на стадии его развития и на этапе эксплуатации платформы. Для перевода платформы в режим полномасштабного функционирования, как правило, необходимо от трех до пяти лет. Второй аспект связан с балансировкой целевых установок проекта, с одной стороны, на проведение научных исследований по созданию платформы, а с другой стороны — на его эксплуатацию для предметных научных исследований. Третий аспект — выбор технологии, где есть три пути: промышленные стандарты; свободно распространяемое программное обеспечение (СПО); собственные разработки. Четвертый аспект — широкое участие научного сообщества в разработке платформы, при котором с большей вероятностью можно реализовать в системе именно то, что ожидают конечные пользователи.

Несмотря на наличие современных технологий, для построения платформ и порталов по-прежнему необходимы значительные затраты человеческих ресурсов. Анализ технологических возможностей, которые традиционно используются при разработке порталов и программно-аппаратных платформ, позволяет выделить общие элементы и проблемы [4], с которыми сталкиваются разработчики таких систем. В одной из работ [5] авторы проанализировали проблему обеспечения коллективов исследовательских центров современными информационными, вычислительными и телекоммуникационными технологическими средствами. В настоящей работе описываются пути реализации предлагаемого авторами подхода.

Анализ существующих порталов и платформ

Проведение моделирования состоит из двух основных частей. Первая — это проведение численных расчетов на вычислительных ресурсах. Вторая — это обработка данных, их представление в удобной форме и их визуализация.

Для возможности удаленной визуализации расчетов, проводимых на распределенной системе TeraGrid [6], был разработан портал TGViz [7] (*TeraGrid Visualization Portal*). Для этого в системе появилась необходимость запуска сервера ParView на вычислительных ресурсах TeraGrid и использование соответствующего ParaView-клиента на рабочей станции пользователя для отображения визуальной информации. Портал TGViz осуществляет контроль за выполнением заданий на TeraGrid и передает пользователю информацию, которая необходима ему для подсоединения к нужному порту и серверу, а также для работы системы визуализации. Портал TGViz построен на системах uPortal [8] и GridSphere [9]. Для запуска задач на TeraGrid используется система GRAM [10]. Такой подход предусматривает возможность реализации удаленного просмотра каталогов, а также удаленного запуска приложений.

В проекте SIDGrid [11] (*Social Informatics Data Grid*) были разработаны портал по информатике и дата-центр для научных сотрудников в области социологии и исследования социального поведения. Система SIDGrid позволяет загружать в нее большие объемы данных, а также предоставляет возможность масштабирования анализа данных до размеров вычислительных ресурсов TeraGrid. Типична ситуация, когда каждый исследователь имеет большой объем хранимой информации и собственные алгоритмы их обработки. При этом у пользователя появляется объективное желание иметь доступ к данным и программному инструментарию коллег по проекту. Важным обстоятельством при реализации такого желания является соблюдение требований приватности. Как следствие, появляется необходимость в тесном взаимодействии разработчиков системы с исследователями-предметниками для интеграции созданного последними программного инструментария в портал и для разработки формата представления выходных данных. Система SIDGrid написана с использованием языков PHP/Drupal [12] и Python [13] для интерфейса пользователя и языка Swift [14] для выполнения вычислительных заданий. Портал SIDGrid демонстрирует эффективные результаты по обработке данных (например, поиск по тегам и образам), по спектру представляемых вэб-сервисов (посредством протокола SOAP) и по возможностям интеграции рабочих станций.

Некоторые порталы используют язык Java для работы с биоинформационными приложениями через интерфейс портала, например, OLSGW (*Open Life Science Gateway*) [15], в котором реализована методика поддержки описания выполнения приложений и их интеграции в портал путем генерации вэб-интерфейса для их исполнения.

Портал Unihub [16], построенный на основе технологии HubZero [17], служит технологической платформой программы "Университетский кластер".

Анализ платформ и порталов, подобных отмеченным выше, позволяет выделить функциональные компоненты, которые встречаются наиболее часто. Пользователи портала хотят контролировать процессы преобразования и идентификации данных, поиска данных и их упорядочения. Как правило, порталы строятся вокруг статической коллекции данных. Пользователи при этом зачастую имеют доступ к разным вычислительным ресурсам и хотят контролировать процесс выполнения задач на всех доступных ресурсах, получая информацию о месте выполнения, состоянии использования приложений, о заданных параметрах и используемых файлах.

Особый интерес представляет для исследователей информационно-вычислительная среда для выполнения научных исследований, доступ к которой реализуется через дружественный интерфейс платформы. С этих позиций важно предоставить инструментарий, позволяющий пользователю сравнительно легко интегрировать доступные ему вычислительные средства и программное обеспечение в необходимую для выполнения конкретного проекта программно-аппаратную среду.

Принципы построения платформы

Подход к построению платформы, который предлагается в настоящей работе, может быть основан на следующих основных принципах [4].

- *Использование проверенных, хорошо разработанных реализаций (СПО).* Выбор зрелых реализаций СПО позволяет обеспечить стабильность работы платформы, а также облегчить устранение ошибок и установку регулярно издаваемых его модификаций в ходе эксплуатации.
- *Улучшение разработки программного кода с помощью проверенных на практике технологий.* Важен выбор технологий, которые дают лучшее качество кода и помогают как на начальном этапе его разработки, так и при внедрении.
- *Важность интуитивных интерфейсов.* Платформа разрабатывается для широкого спектра пользователей с различными техническими навыками. Создание интуитивного интерфейса поможет пользователям работать более продуктивно и без посторонней помощи, что уменьшит давление на разработчиков платформы.
- *Привлечение широкого сообщества для разработки.* Проекты, направленные на активное сообщество, потенциально способны создать компоненты, которые могут быть использованы для дальнейшего улучшения платформы.
- *Большое сообщество разработчиков.* Трудно найти хороших специалистов, и, как следствие, проект платформы будет успешен, если избрать путь независи-

мости от конкретных разработчиков. Необходимо использование технологии, которая имеет широкую поддержку у разработчиков и позволяет увеличить участие в проекте наиболее профессионально подготовленных из них.

Выбор архитектуры и программного обеспечения

В отчете Национального научного фонда (*National Science Foundation*, США) "Революция в науке и инжиниринге посредством киберинфраструктуры" [18, 19] описаны основные направления проведения разработок информационных технологий, необходимых для успешного развития науки. К таким направлениям в первую очередь относится разработка отвечающей самым передовым требованиям единой инфраструктуры, объединяющей: вычислительные ресурсы; сети передачи данных; системы хранения информации, базы данных; прикладное и системное программное обеспечение. Построение такой системы имеет следующие аспекты:

- развитие веб-технологий позволяет исследователям организовать совместную работу над проектом;
- численное моделирование дает возможность исследовать задачи, аналитическое исследование которых затруднено или невозможно на сегодняшний день;
- компьютерное исследование моделей нередко оказывается дешевле, быстрее и проще в реализации, чем экспериментальное исследование физических объектов;
- развитие сетей и технологий дает возможность общения и совместной работы в режиме реального времени;
- наблюдается потребность в обработке больших и перманентно увеличивающихся объемов данных;
- информационные технологии дают новые возможности визуализации результатов исследований.

В контексте работы, результаты которой представлены в настоящей публикации, решалась задача разработки программно-аппаратной платформы, обеспечивающей функционирование веб-ориентированного производственно-исследовательского центра в области вычислительной физики. Такой центр призван объединять современные концепции веб 2.0 (группы пользователей, форумы, вики и др.) и функциональные возможности доступа к прикладным моделям, базам данных, к хранилищам информации и средствам ее визуализации (анимации). В его задачи входят совместная разработка приложений распределенными коллективами, проведение научных исследований в области вычислительной физики на основе этих приложений, а также организация обучения студентов и аспирантов.

В настоящее время разрабатываемая платформа, именуемая КомпФиз, обладает следующими функциональными возможностями:

- обеспечивается доступ конечного пользователя ко всем функциям платформы через веб-портал с использованием возможностей стандартного браузера, в том числе доступ к интерактивным, графическим инструментам моделирования;
- предоставляется возможность визуализации результатов расчетов, в том числе в режиме реального времени;
- реализуются возможности распределенной разработки новых инструментальных средств моделирования, проводимого междисциплинарными коллективами специалистов, и интеграции новых средств моделирования в платформу;
- поддерживаются механизмы интеграции в состав вычислительной инфраструктуры платформы новых аппаратных ресурсов (кластеров, систем хранения данных, систем визуализации и пр.);
- поддерживаются механизмы организации и управления информационной средой взаимодействия групп специалистов ("онлайн-документация", вики);
- реализуются различные методы организации образовательных процессов (интерактивные обучающие курсы, лабораторные работы, виртуальные учебные классы и пр.);
- обеспечивается техническая поддержка пользователей платформы;
- предоставляется возможность совместного доступа пользователей к результатам расчетов;
- предоставляется возможность совместной работы над научной документацией, включая статьи, заметки, комментарии.

Система состоит из следующих подсистем:

- операционная система;
- подсистема управления контентом (CMS), реализующая основные функции веб 2.0, а именно веб-портал, обмен и публикации научных данных, заметок и статей, создание образовательных курсов;
- веб-сервер, выполняющий передачу данных от сервера к пользователям по протоколу HTTP;
- база данных, обеспечивающая хранение данных, создаваемых пользователями, в том числе заметок, обучающих курсов, статей;
- подсистема хранения информации, а именно данных с результатами моделирования, требующих больших объемов памяти (от нескольких мегабайт до терабайт);
- подсистема запуска приложений пользователя (подсистема обработки данных и распределения нагрузки);
- подсистема интеграции платформы со сторонними вычислительными ресурсами;
- подсистема авторизации пользователей;
- подсистема визуализации данных и доступа к графическому интерфейсу приложений пользователей;
- подсистема для совместной разработки приложений;
- подсистема мониторинга состояния аппаратных составляющих платформы;
- подсистема управления программным комплексом платформы.

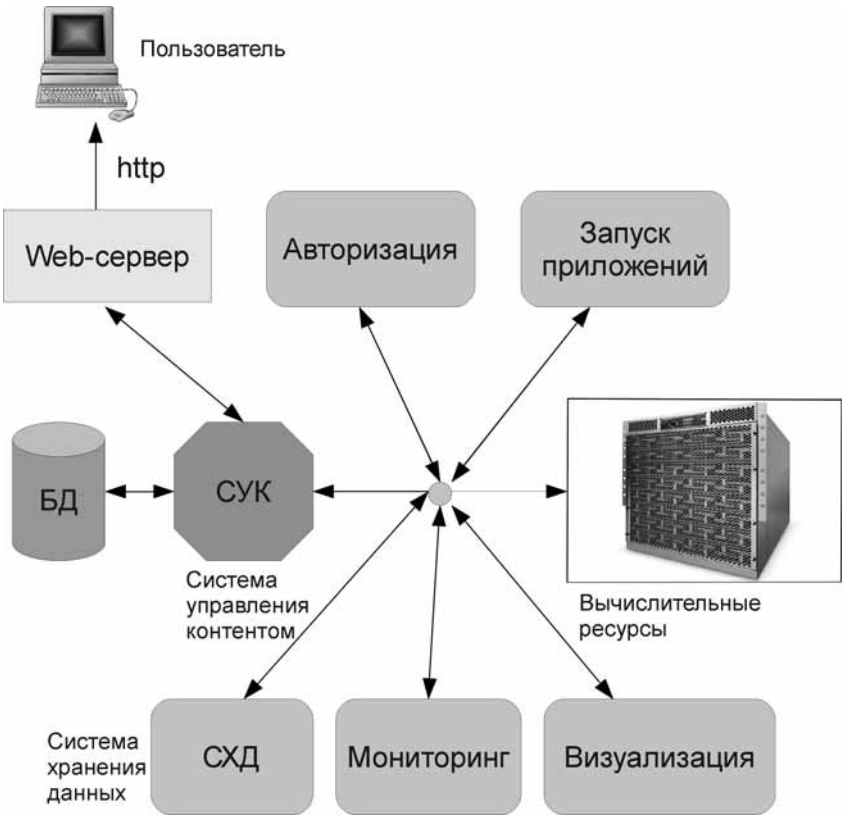


Рис. 1. Архитектура веб-ориентированной платформы

Схематично архитектура платформы представлена на рис. 1.

Использование в системе СПО дает большое преимущество. Так, пользователь может сам доработать его коды под свои нужды или использовать программный код, разработанный другими авторами, как часть своего продукта. Обмен программными наработками похож на обмен научными идеями, любой человек, при наличии такой возможности, способен воспользоваться плодами работы других людей в своей деятельности. Это существенно упрощает разработку, так как наиболее популярные и востребованные программные продукты уже реализованы. В качестве операционной системы в рассматриваемой платформе используется ОС Linux. В качестве подсистемы, выполняющей роль базы данных, может быть выбрана любая из реализаций баз данных с интерфейсом SQL. В частности, авторами были рассмотрены MySQL и PostgreSQL. Производительность и возможности обеих систем практически одинаковы и выбор в пользу одной из них является вопросом личных предпочтений.

Подсистема, выполняющая функции CMS — системы управления контентом, является ядром системы. Она поддерживает функции связующего звена, объединяющего все компоненты в единое целое. Система должна быть легко расширяемой, удобной в использовании, безопасной и производительной. Систем, обладающих подобным функционалом и распространяемых под открытой лицензией, несколько десятков. Среди наиболее популярных можно выделить Joomla, Drupal и Wordpress. На сайте http://extensions.joomla.org/ представлено более 8000 дополнений, что является неоспоримым преимуществом этой системы, так как позволяет реализовать значительную часть функциональных возможностей платформы без затрат дополнительных ресурсов на их разработку.

Подсистема хранения информации должна обеспечивать хранение данных большого объема, доступ к данным на высокой скорости, надежность хранения данных. Обработка данных проводится с использованием вычислительного кластера, когда несколько приложений, запущенные на разных узлах, могут одновременно считывать и записывать данные. В качестве возможных решений можно привести сетевые файловые системы NFS, AFS, распределенные файловые системы GlusterFS, GFS, Lustre и др. Наиболее простой в использовании является система NFS, поскольку ее реализация входит во все стандартные дистрибутивы Linux. Однако как показывает практика, при проведении больших вычислений с сильной зависимостью их результатов от данных, использование данной системы может стать "узким местом". Система NFS не обеспечивает также надежного хранения, а полагается целиком на надежность нижележащего блочного устройства.

Файловые системы, такие как Lustre и GFS, предполагают практически обязательное использование дополнительной аппаратной инфраструктуры в виде сети передачи данных, включающей высокопроизводительное сетевое файловое хранилище с интерфейсом FiberChannel или iSCSI. В рамках рассматриваемого проекта использование подобных промышленных систем не оправдано в силу их высокой стоимости и сложностей, которые возникают при эксплуатации. Альтернатива подобным системам — программная система GlusterFS. Она использует сеть на базе протокола TCP/IP и традиционные файловые системы (ext3, ext4) для объединения ресурсов нескольких узлов в единое целое. При этом поддерживаются механизмы репликации данных на один или несколько узлов. Эксплуатация этой системы на вычислительном кластере Wall [20] в Научном центре Черноголовка (НЦЧ РАН) показала хорошие результаты производительности, превосходящие возможности традиционной NFS.

Подсистема запуска заданий должна предоставлять возможности по установке и запуску программ, распространяемых в исходных кодах. Большая часть существующего научного программного обеспечения создана с использованием языков программирования

C, C++, Fortran. Практически все версии Linux поставляются в комплекте со свободно распространяемым компилятором GCC. К сожалению, поддержка языка программирования Fortran реализована в GCC не полностью. Для компиляции программ, написанных на языке Fortran 95, или при необходимости задействования особых технологий распараллеливания (таких как SSE4, OpenMP), предпочтительным является использование проприетарного продукта, например, компилятора Intel C Compiler.

Если задание является "простым", например, построение графика, то такая задача может быть запущена на том же узле, что и интерфейс пользователя системы. При этом система управления контентом (СУК) может выступить в роли подсистемы запуска заданий и дополнительных компонентов в этом случае не понадобится.

При необходимости запуска большого числа различных продолжительных заданий должна быть предусмотрена возможность запуска задач на вычислительных узлах кластера. Для этого на кластере обычно используется система очередей совместно с планировщиком заданий. Примерами таких систем являются продукты Torque, OpenPBS, Maui, LSF, Condor, Cleo и др.

Система очередей Torque совместно со свободно распространяемым планировщиком Maui за время эксплуатации на кластерах НЦЧ РАН показали большие возможности по управлению кластером, разграничению и установке приоритетов пользователей, по простоте управления и надежности.

Подсистема интеграции с вычислительным ресурсами должна представлять собой интерфейс, осуществляющий взаимодействие системы СУК с системой очередей, исполняющей непосредственный запуск заданий. В рамках проекта HubZero существуют разработанные модули, реализующие необходимый функционал.

Подсистема авторизации и аутентификации пользователей должна обеспечивать их прозрачную аутентификацию в рамках всех используемых подсистем. Это означает, что один идентификатор пользователя должен быть доступен всем другим подсистемам с единым паролем. Поскольку платформа состоит из большого числа разнородных компонентов, необходим выбор универсального протокола. Таким протоколом является LDAP — легкий протокол доступа к каталогам. Возможности авторизации пользователей посредством службы каталогов LDAP есть у значительного числа прикладных программ. Свободно распространяемая реализация данного протокола носит название openldap.

Подсистема визуализации должна обеспечивать доступ пользователя к графическим данным, в частности, к графическому интерфейсу прикладных программ, запущенных в вычислительной среде, предоставляя пользователю аналог удаленного рабочего стола. Такая возможность должна быть доступна через веб-интерфейс системы. Для ОС Linux существует реализация протокола VNC (*Virtual Network Client*), входящая в состав графического сервера системы X.org X11Vnc. Для отображения данных посредством веб-интерфейса должен быть использован Java-апплет, реализующий данный функционал на стороне клиента. В рамках проекта HubZero разработаны новые и адаптированы уже существующие модули с открытым программным кодом, реализующие необходимые функциональные возможности.

Также подсистема визуализации должна включать возможность вывода изображений сверхвысокого разрешения с помощью мозаичной видеостены. Открытый программный продукт, реализующий такую функцию, создан в университете Иллинойса в Чикаго и носит название SAGE (*Scalable Adaptive Graphics Environment*) — масштабируемая адаптивная графическая среда [21]. Программная система SAGE предоставляет возможность строить видеостены с помощью программных средств, позволяющих объединить разнородные мониторы, подключенные к разным узлам, в единое целое. Разработка двух подсистем визуализации — посредством протокола VNC и с помощью технологии SAGE — позволяет реализовать механизмы как полностью удаленной визуализации, так и локальной, с использованием аппаратных ресурсов, входящих в комплект вычислительного кластера.

Подсистема совместной разработки поддерживает механизмы хранения данных и исходных текстов программ, обеспечивающие к ним конкурентный доступ, позволяет фиксировать все модификации кода и, как следствие, осуществлять контроль версий программ. В рамках СПО существуют многочисленные реализации подобных систем, например, CVS, SVN, GIT и др. В последнее время наметилась тенденция перехода на систему GIT, которая позволяет вести распределенную разработку данных. Однако поскольку в рамках рассматриваемого в настоящей работе проекта вся разработка должна вестись посредством веб-интерфейса на сервере системы, а не на локальных компьютерах пользователей, возможности SVN также удовлетворяют предъявляемым к подобной системе требованиям.

Подсистема мониторинга должна обеспечивать возможность сбора и визуального отображения информации. При этом должна отображаться информация о состоянии системы в целях анализа загруженности вычислительных узлов и каналов передачи данных, а также о наличии аномальных ситуаций в используемом оборудовании. Среди систем с открытым исходным кодом, выполняющих эти функции, на IT-рынке существуют: Nagios, Ganglia, Zabbix, Zenoss, а также ряд других продуктов.

Система Nagios развивается продолжительное время, есть большое число разработанных к ней дополнений, однако ее главный недостаток — высокая сложность конфигурирования системы. Система Ganglia используется преимущественно для мониторинга загрузки вычислительных узлов. Вместе с тем в ней отсутствуют встроенные средства оповещения о нерегламентированных событиях. Системы Zabbix и Zenoss — программные средства одного уровня, обеспечивающие как единую консоль, отображающую общее состояние

системы, различные средства рассылки уведомлений, возможность расширения дополнительными модулями и многое другое.

Подсистема управления платформой должна поддерживать механизмы создания новых пользователей в системе, возможности по управлению правами доступа к ее ресурсам, возможность контроля запущенных задач. Часть необходимых для этого функций планируется реализовать в рамках подсистемы СУК, выбранной для эксплуатации. Рассматривается также возможность управления системой в интерактивном режиме с помощью консольных утилит и с использованием разработанных для этого скриптов, автоматизирующих часть повседневных действий администратора системы, таких как добавление пользователей.

Для реализации рассматриваемой платформы принята свободно распространяемая система Hubzero, включающая в себя многие необходимые компоненты. Для реализации недостающих возможностей выбраны следующие программные средства:

- Zenoss — для организации подсистемы мониторинга;
- SAGE — для подсистемы визуализации изображений сверхвысокого разрешения;
- Gluster — для организации высокопроизводительного и высоконадежного файлового хранилища;
- Torque и Maui — для организации доступа к вычислительным ресурсам кластера.

Система Hubzero поставляется либо в виде набора пакетов для ОС Debian, либо в качестве образа виртуальной машины (VM) с предустановленной ОС Debian и всеми компонентами. Использование готового образа виртуальной машины позволяет упростить начальную установку системы. Однако в дальнейшем, для достижения лучшей производительности, может потребоваться отказ от системы виртуализации и установка ОС Debian и Hubzero на выделенную управляющую машину.

Для использования программного обеспечения Hubzero в качестве основы платформы системы потребовалась адаптация компонентов Hubzero с учетом особенностей существующих аппаратных ресурсов.

Для пилотной версии были выбраны реализации инструментов моделирования таких физических задач, как ограниченная диффузией агрегация (DLA — *Diffusion Limited Aggregation*) [22], базовая модель ферромагнетика (модель Изинга, *Ising*) [23] и модель испарения лежащей на подложке капли капиллярного размера [24]. В целях обеспечения обработки результатов исследований и подготовки публикаций в систему были интегрированы пакеты Qtiplot, Openoffice, TeX.

На рис. 2 представлена общая схема платформы КомпФиз [25].

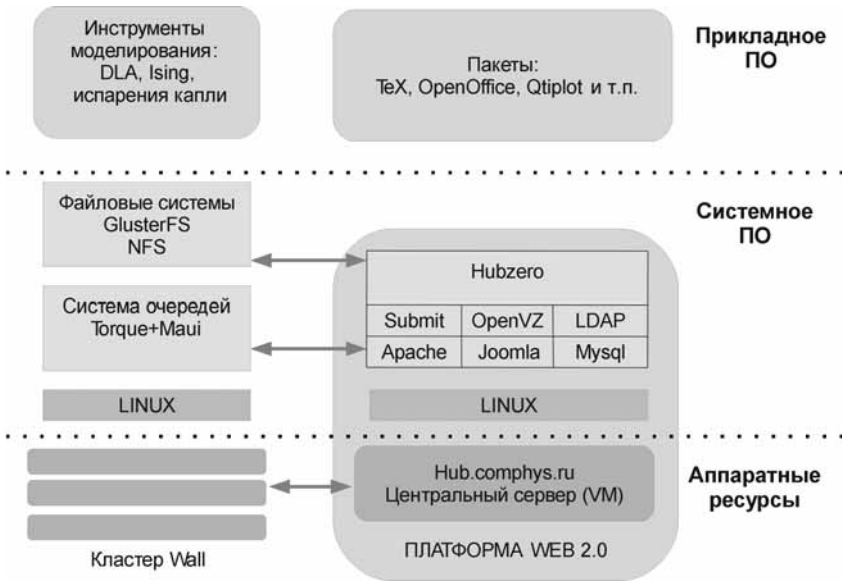


Рис. 2. Общая схема платформы КомпФиз

Заключение

В настоящей работе описана архитектура и реализация программно-аппаратной платформы для проведения научных исследований в области вычислительной физики. Построенная платформа может быть использована как для проведения научных исследований, так и для организации учебного процесса. Важно, что пользователь-предметник имеет в своем распоряжении большой набор распределенных вычислительных и информационных ресурсов, задействовать которые он может, не обладая специальными знаниями в области инфокоммуникационных систем. Технология доступа к ресурсам хранения и обработки информации и проведения вычислительных расчетов скрыта за удобным и интуитивно понятным интерфейсом.

В настоящей работе принят подход на основе использования СПО [23], который позволяет за сравнительно небольшое время интегрировать выбранные программные системы в единую программно-аппаратную среду и адаптировать их под решение конкретного набора задач.

При разработке платформы, по мнению авторов, удалось учесть три из четырех аспектов устойчивости развития порталов и платформ, отмеченных во введении. Для эффективного внедрения разработанной платформы требуется устойчивое финансирование в течение нескольких лет, что отмечено в первом аспекте. Таким образом, успешное развитие платформ и порталов по естественным наукам зависит от источников финансирования, направленных на поддержку и внедрение инфраструктур, разработанных по другим программам и грантам. Наличие инфраструктурных проектов в области инфокоммуникационных систем, по мнению авторов, является ключевым для успешного развития научной и учебной деятельности.

Настоящая работа выполнена по проекту, выполняемому в рамках ФЦП "Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007—2013 годы", государственный контракт 07.514.11.4041 с Министерством образования и науки.

Список литературы

1. **Foster I., Kesselman C., Tuecke S.** The Anatomy of the Grid: Enabling Scalable Virtual Organization // International Journal of Supercomputer Applications. 2001. Vol. 15. N 3. P. 200—222.

2. **Armbrust M., Fox A.** et al. View of Cloud Computing // Communications of the ACM 53. 2010. April. P. 50—58.

3. **Wilkins-Diehr N. and Lawrence K. A.** Opening Science Gateways to Future Success: The Challenges of Gateway Sustainability // Gateway Computing Environments Workshop (GCE). 2010. P. 1—10. DOI: 10.1109/GCE.2010.5676121

4. **Uram T. D., Papka M. E., Hereld M., Wilde M.** A solution looking for lots of problems: generic portals for science infrastructure // Proc. of the 2011 TeraGrid Conference: Extreme Digital Discovery, ACM Digital Library. URL: http://dl.acm.org/citation.cfm?id=2016741

5. **Щур Л. Н., Меньшутин А. Ю., Шикота С. К.** Инфокоммуникационное обеспечение исследовательского центра: задачи и инфраструктура // Информационное общество. 2011. № 6. С. 58—65.

6. https://www.xsede.org/

7. **Insley J. A., Papka M. E.** Prototyping Simple Access to Visualization Resources. URL: http://www.globus.org/alliance/publications/clusterworld/0605Grid.pdf

8. **Проект** uPortal. URL: http://www.jasig.org/uportal/

9. **Проект** GridSphere. URL: http://www.vlab.msi.umn.edu/gridsphere/gridsphere

10. **Проект** GRAM. URL: http://dev.globus.org/wiki/GRAM

11. **Levow G.-A., Waxmonsky S., Bertenthal B.** et al. SIDGrid: A Framework for Distributed, Integrated Multimodal Annotation, Archiving, and Analysis. URL: http://anl.academia.edu/MarkHereld/Papers/828780/SIDGrid_A_Framework_for_Distributed_Integrated_Multimodal_Annotation_Archiving_and_Analysis

12. **Проект** Drupal. URL: http://drupal.org/

13. **Проект** Python. URL: http://www.python.org/

14. **Wilde M., Hategan M., Wozniak J. M.**, **Clifford B., Katz D. S., Foster I.** Swift: A language for distributed parallel scripting // Parallel Computing. 2011. № 39. P. 633—652.

15. **Wu W., Edwards R., Judson I. R.** et al. TeraGrid Open Life Science Gateway // TeraGrid 2008 conference. Las Vegas. June 9—13, 2008. P. 97—105.

16. **Технологическая** платформа программы "Университетский кластер". URL: http://unihub.ru

17. **McLennan M., Kline G.** HUBzero Paving the Way for the Third Pillar of Science // HPC in the Cloud. 2011. № 9. P. 11—18.

18. **Atkins D. E.** et al. Revolutionizing Science and Engineering Through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure. URL: http://www.nsf.gov/od/oci/reports/toc.jsp

19. **Cyberinfrastructure** Vision for 21st Century Discovery. National Science Foundation Cyberinfrastructure Council. URL: http://www.nsf.gov/pubs/2007/nsf0728/index.jsp

20. **Григорьева М. В., Крашаков С. А., Меньшутин А. Ю., Разумов В. Ф., Шикота С. К., Щур Л. Н.** Высокопроизводительный программно-аппаратный комплекс для приема и отображения визуальной информации // Труды суперкомпьютерного форума "Суперкомпьютерные технологии в образовании, науке и промышленности". Нижний Новгород, 1—3 ноября, 2011 г. Изд-во ННГУ им. Н. И. Лобачевского. 2011. С. 96—99.

21. **Jeong B., Leigh J., Johnson A.** et al. Ultrascale Collaborative Visualization Using a Display-Rich Global Cyberinfrastructure // IEEE Computer Graphics and Applications. 2010. № 30. P. 71—83.

22. **Menshutin A. Yu., Shchur L. N.** Morphological diagram of diffusion driven aggregate growth in plane: Competition of anisotropy and adhesion // Computer Physics Communications. 2011. Vol. 182. P. 1819—1823.

23. **Щур Л. Н.** Вычислительная физика и проверка теоретических предсказаний // Успехи физических наук. 2012. Т. 182. С. 787—792.

24. **Barash L. Yu., Bigioni T. P., Vinokur V. M., Shchur L. N.** Evaporation and fluid dynamics of a sessile drop of capillary size // Physical Review E. 2009. 79.P.046301.1-16.

25. **Производственно-исследовательский** центр в области вычислительной физики КомпФиз. URL: http://hub.comphys.ru

26. **Иванников В. П.** Что такое СПО // Сб. трудов Перспективные компьютерные системы: устройства, методы и концепции / под ред. Р. Р. Назирова и Л. Н. Щура. 2011. ИКИ РАН, Москва. С. 106—114.

ИНФОРМАЦИЯ

УДК 656.073 + 656.22; 004.75

А. Б. Шабунин[1], руководитель научно-технического комплекса проектирования и разработки информационных систем, e-mail: a_shabunin@hotmail.com,
С. Н. Марков[2], технический директор,e-mail: markovs@programpark.ru,
Д. В. Дмитриев[2], нач. отдела, e-mail: dmtrvdnl@gmail.com,
Н. А. Кузнецов[3], д-р техн. наук, советник, e-mail: kuznetsov@cplire.ru,
П. О. Скобелев[4, 5], д-р техн. наук, вед. науч. сотр., ген. директор, e-mail: petr.skobelev@gmail.com,
С. С. Кожевников[6], директор аналитического центра, e-mail: Kozhevnikovss@gmail.com,
Е. В. Симонова[6], канд. техн. наук, вед. аналитик, e-mail: Simonova.Elena.V@gmail.com,
А. В. Царев[6], ген. директор, e-mail: at@anarun.net
[1]ОАО "НИИАС", г. Москва,
[2]ООО "ПрограмПарк", г. Москва,
[3]Институт радиотехники и электроники им. В. А. Котельникова РАН, г. Москва,
[4]Институт проблем управления сложными системами РАН, г. Самара,
[5]ООО "НПК "Мультиагентные технологии", г. Самара,
[6]ООО "НПК "Разумные решения", г. Самара

Èíòåãðàöèîííàÿ ïëàòôîðìà äëÿ ðàñïðåäåëåííûõ èíòåëëåêòóàëüíûõ ñèñòåì óïðàâëåíèÿ æåëåçíîäîðîæíûì òðàíñïîðòîì ÎÀÎ "ÐÆÄ"

Описываются принципы реализации интеграционной платформы для распределенных интеллектуальных систем управления железнодорожным транспортом ОАО "РЖД" на основе сетецентрического подхода. Рассматриваются структура и реализация интеграционной платформы, отраслевой язык программирования DSL. Представлен прототип интеграционной платформы, обеспечивающий реализацию автоматизированных систем управления движением на направлении Санкт-Петербург — Москва полигона "Октябрьская дорога".

Ключевые слова: *сетецентрический подход, интеллектуальная система, интеграционная платформа, онтология, отраслевой язык*

Введение

Эффективное управление железнодорожным транспортом ОАО "РЖД", одной из крупнейших национальных компаний, не может обеспечиваться одной централизованной системой, равно как и множеством отдельных систем, не имеющих между собой надежных связей, единых алгоритмов и стандартов. Под ресурсами здесь и далее подразумеваются: подвижной состав (транспортные средства), железнодорожные сети, включая технические средства управления транспортными потоками.

При решении сложных задач автоматизации управления железнодорожным транспортом, включающих обеспечение производственных, организационных и технологических составляющих перевозочного процесса, представляется целесообразным использовать

сетецентрический подход. Такой подход получил развитие для моделирования боевых действий с подразделениями различных родов войск в крупных боевых операциях [1, 2], в ходе которых обеспечивается согласованная работа всей сети поддерживающих такие действия средств и систем управления: "так локально, как только возможно, и так глобально, насколько это требуется для успешного решения задачи" (*as local as possible and as global — as required*).

Сетецентрический подход к управлению сложными системами

Сетецентрический подход предопределяет организацию управления сложными процессами в распределенной информационно-коммуникационной инфраструктуре. Такая инфраструктура реализует максимальную ситуационную осведомленность (*Situational Awareness*) каждого из ее информационных узлов (далее — узлов) и переход к их работе в режиме самоорганизации на достижение задач, поставленных на каждом из иерархических уровней управления. Успешное решение задач управления всей сложной системой в рамках сетецентрического подхода складывается из успешного выполнения своей задачи каждым узлом во взаимодействии с другими смежными узлами. В результате взаимодействия узлов друг с другом происходит динамическое (изменяющееся во времени) согласование действий каждого из них под контролем особого узла (диспетчера командного штаба), имеющего стратегические сведения и результаты анализа ситуации в целом. Эти узлы, в свою очередь, также могут образовывать связанную сеть взаимодействующих узлов [3].

Основу сетецентрического управления составляют сквозные модели систем связанных объектов, взаимодействующих в едином информационном пространстве, в котором в реальном времени и с высокой надежностью обеспечивается циклическое повторение всех этапов исполнения различных контуров управления:

- многоканальный сбор, первичная обработка и накопление разноплановой фрагментарной информации, поступающей с различных компонентов системы, отражающих состояние внешней среды и их текущее внутреннее состояние;
- формирование (на основе результатов многоуровневой обработки собираемой и накапливаемой информации) целостной картины из информационных фрагментов, которая определяет текущее состояние системы в целом и отдельных ее частей, являясь основой для выработки управляющих воздействий;
- выработка и доставка управляющих воздействий, структурированных при формировании общей модели таким образом, что управляющая информация каждого такого канала воздействия выстроена в соответствии с "компетенцией" соответствующего управляемого компонента [4].

Для реализации сетецентрического подхода в системах управления железнодорожным транспортом "ОАО "РЖД" используется интеграционная платформа, соответствующая принципам, заложенным данным подходом. Она обеспечивает единую информационную среду для автоматизации процессов поддержки принятия решений по планированию, выполнению и контролю сквозных производственных процессов [5—7].

Интеграционная платформа

На основе интеграционной платформы осуществляется взаимодействие с уже существующей информационной инфраструктурой ОАО "РЖД", а также поддерживается среда проектирования для отраслевых экспертов и разработчиков, которые с ее помощью смогут создавать новые, существующие в едином информационном пространстве и взаимодействующие между собой подсистемы, ориентированные на выполнение различных функций. Такая платформа содержит набор программно-аппаратных средств, компонентов, отраслевых стандартов, шаблонов и соглашений по проектированию. Она ориентирована на комплексную автоматизацию производственных процессов и позволяет создавать распределенные системы произвольной топологии.

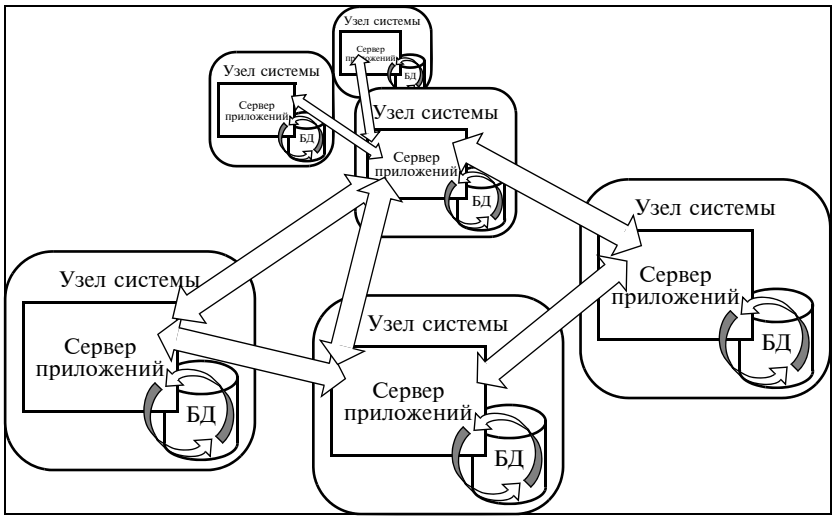


Рис. 1. Компонентная структура платформы и организация распределенной системы

Каждый узел, согласно топологии интеграционной платформы, состоит из набора программно-аппаратных средств, включающих в качестве основного компонента резервируемый сервер приложений, а также опциональную реляционную или объектную базу данных. Все функциональные возможности платформы реализуются сервером приложений. Компонентная структура платформы и взаимодействие между узлами представляющей ее распределенной системы управления показаны на рис. 1.

Программная архитектура одного узла системы управления приведена на рис. 2.

Взаимодействие между функциональными модулями отражено стрелками с нумерацией.

На базе *программной платформы общего назначения Вектор М* формируется (*1*) внутренний *язык программирования платформы* (DSL-I). Внутренний язык обеспечивает конструкции, позволяющие формализовать и использовать шаблоны проектирования систем. В их число входят классы, модули, методы, объекты, фильтры, решающие правила, регуляторы, коммутаторы, технологические и бизнес-процессы, модели, домены, события и сообщения, машины состояний, узлы топологии. Следует добавить также большое число элементов создания пользовательского интерфейса.

На базе системного языка и внутреннего языка платформы Вектор М строится (*2*, *3*) *онтология предметной области*. Онтология является единой для всей системы и доступна в каждом ее узле. Каждое изменение онтологии распространяется по всей сети узлов системы.

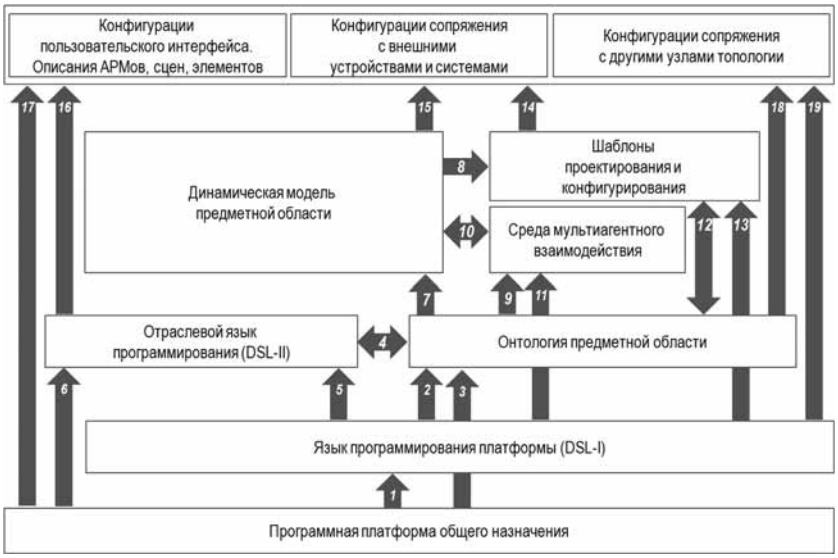


Рис. 2. Структура сервера приложений платформы

Онтология, *как семантическая модель,* выступает основой построения *предметно-ориентированного языка* программирования железнодорожной отрасли, так называемого *отраслевого языка* (*4*) (DSL-II). Отдельные элементы онтологии могут быть реализованы программно с применением конструкций *отраслевого языка* [8]. Возможности отраслевого языка существенно увеличиваются, если он является продолжением внутреннего системного языка платформы (*5*) и языка общего назначения (*6*). Таким образом, для программиста-технолога доступны несколько уровней языков программирования, которые он может использовать при разработке функциональных модулей системы.

Высокая степень адаптивности поведения системы реализуется за счет построения динамической модели предметной области на основе онтологии (*7*). Динамическая модель образуется объектами предметной области с изменяемым набором характеризующих их атрибутов и поведением — отношениями между собой и со средой окружения. Все множество объектов организовано по различным контекстам — пространствам данных, подсистемам и хранилищам.

Функциональные задачи не программируются в традиционном понимании, а проектируются на множестве имеющихся *шаблонов проектирования и конфигурирования* с использованием отраслевого и базового языков. Платформа позволяет создавать и расширять не только отраслевой язык, но и набор шаблонов проектирования. Шаблоны строятся (*13*) с использованием языка программирования платформы и на основе выборки (*8*) объектов из общего множества. Часто используемым шаблоном проектирования является шаблон под названием *модель.* Шаблон модель представляет собой специализированную структуру данных, например, простой список, древовидный список, хэш-таблицу, простую таблицу и др. Шаблон модель позволяет эффективно проводить конфигурирование (*14*) бизнес-логики, пользовательский интерфейс и сопряжение с другими системами и устройствами. К реализованным шаблонам проектирования также относятся *машина состояний, домен, технологический процесс* и др. Методология проектирования на платформе Вектор М рекомендует максимально возможное использование шаблонов (*12*) в качестве крупных строительных блоков при построении как онтологии предметной области при реализации систем, так и реализации таких аспектов системы как пользовательский интерфейс.

В состав платформы встраивается модуль *мультиагентного взаимодействия*, реализующий функции динамического планирования и оптимизации ресурсов. Модуль реализует распределенное сообщество *агентов*-планировщиков (оптимизаторов) с собственными расписаниями, действующих параллельно и на основе переговоров. Агенты синхронизированы (*9*, *10*) с динамическими объектами предметной области и конфигурируются (*11*) с использованием внутреннего языка программирования платформы.

Конфигурация пользовательских интерфейсов реализуется на внутреннем языке системы (*17*), предоставляющем возможности описания элементов пользовательского интерфейса и построения из них сцен и приложений, исполняемых в едином информационном пространстве. Внутренняя логика приложений проектируется с использованием конструкций отраслевого языка (*16*). Возможности пользовательских приложений существенно расширяются за счет использования языка платформы (*19*) и базового языка программирования. Использование шаблонов проектирования (*14*) при конфигурировании пользовательских приложений существенно снижает сложность разработки, привносит модульность и позволяет разрабатывать и поддерживать методологии проектирования.

Система получает информацию о реальной ситуации из множества разнородных источников. Платформа обеспечивает механизм аккумуляции первичной информации и трансформацию разнородной первичной информации в унифицированное внутреннее представление (*15*). Задача *сопряжения с внешними устройствами и системами* решается унифицированным в рамках системы подходом — за счет создания и конфигурации соответствующих моделей взаимодействия и трансформации данных.

Конфигурация сопряжения с другими узлами топологии определяет наборы событий и сущностей (*18*), по которым узел собирается оповещать другие узлы системы. Эти наборы могут быть представлены как перечень пересечений и(или) объединений типов, различным образом организованных контекстов или политик, либо как перечень моделей. Платформа обеспечивает синхронизацию представления (реализации) онтологии на всех узлах системы.

Технология проектирования

Проектирование и разработка крупных автоматизированных систем логически сложная, трудоемкая и длительная работа, требующая высокой квалификации участвующих в ней специалистов. Особенно это касается масштабных проектов с распределенной топологией и необходимостью интеграции разнородного программного обеспечения. В текущем разделе приводится описание особенностей проектирования с применением инструментария и технологии проектирования автоматизированных систем на основе интеграционной платформы.

Платформа обеспечивает одновременно среду разработки (проектирования и конфигурирования) системы и среду выполнения. Основным исполнителем работ по созданию систем на базе платформы выступает программист-технолог или проектировщик. Он создает систему на отраслевом языке, следуя шаблонам проектирования, которые предлагает платформа.

Неотъемлемой частью платформы является наличие автоматизированного рабочего места проектировщика (АРМ "Конфигуратор") для создания и конфигурирования приложений — рабочих мест пользователей и задания логики функционирования системы.

На рис. 3 (см. третью сторону обложки) представлен интерфейс АРМ "Конфигуратор", который содержит набор элементов для работы с сущностями платформы, их экземплярами, а также дополнительные сервисы. Ниже приводится перечень и краткое описание его панелей.

Панель Навигация предназначена для перемещения по сущностям. Панель представляет собой дерево сущностей и групп сущностей. При выборе сущности, список соответствующих экземпляров будет выведен на панели Список. Из списка можно "перетаскивать" элементы в панель Редакторы, предназначенную в основном для работы с DSL-скриптами.

В эту панель выводятся также редактор объектов и редактор технологических процессов (при работе с данными сущностями).

Панель DSL-помощник предназначена для получения информации о DSL-командах, которые выполняются при создании, редактировании или удалении экземпляров сущностей. В данной панели представлено описание и программный код команды, который переносится в DSL-редактор, обеспечивая полуавтоматический набор кода. Разным сущностям соответствуют разные DSL-помощники.

Панель Диаграмма предназначена для получения графической информации о взаимосвязи экземпляров сущностей системы, например, диаграмма классов и включенных модулей.

Панель Консоль предназначена для получения информации о выполнении операций в системе, а также для просмотра служебной информации о работе приложения.

Пример интеграционной платформы ОАО "РЖД"

Рассмотрим пример интеграционной платформы, в рамках которой должно быть обеспечено объединение комплексной автоматизации производственных бизнес-процессов, существующих промышленных систем, систем управления ресурсами подразделений железной дороги в реальном времени и доступ к их данным.

На рассматриваемом полигоне "Октябрьская дорога" (Москва — Санкт-Петербург) протяженностью 648 км решается задача управления высокоскоростным пассажирским и пригородным движениями. Управление движением поездов осуществляется на сетевом, дорожном и станционном уровнях.

В данном примере представлен вариант интеграционной платформы, обеспечивающий реализацию автоматизированных систем управления движением. Интеграционная платформа включает базовый уровень функциональности и набор шаблонов и инструментальных средств для реализации модулей системы.

Поддержка распределенных систем. Программно-аппаратная часть платформы реализует распределенную систему на трех уровнях сети узлов топологии:

- первый уровень — сетевой;
- второй уровень — дорожный;
- третий уровень — станционный.

Каждый узел топологии обеспечен каналами поступления и передачи информации, которые реализуют взаимодействие с внешней средой и другими узлами. Активность каждого узла определяется набором динамических ключевых параметров, соответствующих уровню узла и его месту в производственной цепочке, в контексте которой узел получает на вход результаты работы предыдущего узла и передает следующему результаты собственной работы.

В автоматизированной системе на основе базовых решений платформы конфигурируются потоки дан-

ных между уровнями. Примером таких данных на сетевом уровне могут быть плановые и регулировочные воздействия на уровне объемных показателей поездопотоков, передачи поездов между дорогами на стыковых станциях и др. На дорожном уровне объемные показатели детализированы и учитываются в графике движения поездов. График движения является источником для формирования планов приема и отправления поездов на станционном уровне. Исполнение же планов движения поездов формирует обратный поток фактических показателей с заданной детализацией для каждого уровня, которые используются в процессах контроля, анализа и регулирования движения путем своевременного перепланирования.

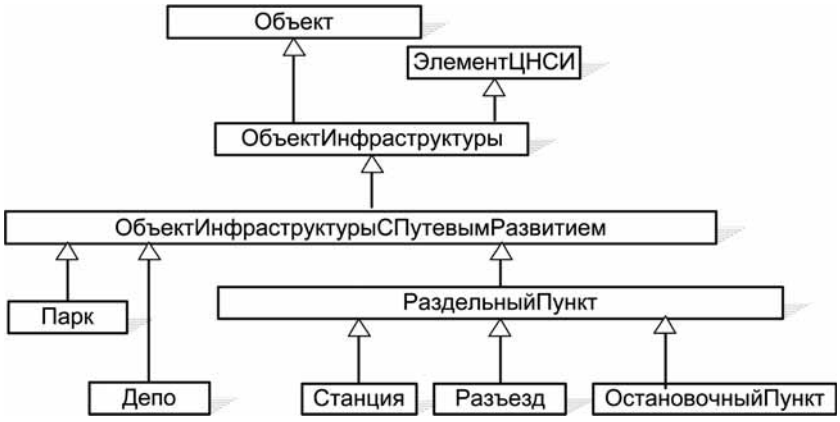


Рис. 4. Диаграмма наследования фрагмента онтологии

Поддержка единой онтологии и отраслевого языка. Онтология в интеграционной платформе представляет собой формализованное описание предметной области в виде классов, модулей, хранилищ объектов и связей между ними. В качестве примера рассмотрим фрагмент онтологии, описывающий некоторые элементы инфраструктуры ОАО "РЖД".

Стандарты и соглашения, используемые при формировании онтологии крупных систем, соответствуют основным принципам объектно-ориентированного проектирования и обеспечивают согласованность модели и отсутствие дублирования характеристик понятий. На рис. 4 представлена диаграмма наследования рассматриваемого фрагмента онтологии.

Для каждого элемента онтологии формируется набор языковых конструкций, обеспеченных функциональными методами поведения объектов. Совокупность данных конструкций реализует словарь отраслевого языка, применяемый при разработке автоматизированных систем в интеграционной платформе. Пример конструкций отраслевого языка для концепта Станция приведен в таблице.

Поддержка отображения динамической объектной модели. Компоненты интеграционной платформы, отвечающие за визуализацию, включают подготовленные шаблоны для настройки в проектируемых приложениях следующих функций отображения:

- объектов инфраструктуры полигона;
- движения поездов;
- графика движения поездов;
- состояния и режимов станций;
- предупреждений и ремонтных окон;
- информации о помехах и нештатных ситуациях;
- экранных форм приложений и диалоговых окон

автоматизированных рабочих мест проектируемых систем.

Пример форм отображения представлен на рис. 5 (см. третью сторону обложки).

Поддержка интеграции с внешними системами. В качестве примера реализации функций по интеграции с внешними системами приводится компонент инте-

Пример конструкций отраслевого языка

Языковая конструкция	Результат выполнения
добавить_маршрут	Добавление маршрута по станции
получить_диспетчерский_участок	Получение диспетчерского участка станции
получить_маршруты	Получение маршрутов по станции
получить_основной_код	Получение основного кода станции
получить_полное_название	Получение полного названия станции
получить_тип	Получение типа станции
удалить_маршрут	Удаление маршрута по станции
установить_диспетчерский_участок	Установка диспетчерского участка станции
установить_основной_код	Установка основного кода станции
установить_полное_название	Установка полного названия станции
установить_тип	Установка типа станции
установлен_диспетчерский_участок?	Проверка, установлен ли диспетчерский участок станции (да/нет)
установлен_основной_код?	Проверка, установлен ли основной код станции (да/нет)
установлен_тип?	Проверка, установлен ли тип станции (да/нет)
установлено_краткое_название?	Проверка, установлено ли краткое название станции (да/нет)
установлено_полное_название?	Проверка, установлено ли полное название станции (да/нет)

грационной платформы, обеспечивающий функции подключения к внешним источникам данных, таким как данные о поездном местоположении, режимах станций и переездов, маршрутах движения поезда по станции, данные о ремонтных окнах и предупреждения об ограничении скорости.

Компонент сопряжения включает следующий набор функций для работы с внешними источниками данных:

- подключение к источникам данных;
- мониторинг доступности источников данных;
- получение информации от источников данных;
- фильтрация информации от источников данных;
- контроль целостности информации от источников данных;
- запись информации от источника данных;
- воспроизведение данных, полученных от внешнего источника;
- преобразование информации от источников данных в элементы динамической объектной модели;
- синхронизация динамической объектной модели с источниками данных.

Заключение

Прототип интеграционной платформы реализуется на базе программно-технического комплекса для испытания информационно-управляющих технологий на направлении железной дороги Санкт-Петербург — Москва.

В ходе разработки прототипа интеграционной платформы были достигнуты следующие результаты:

- сформирована распределенная система, включающая три уровня узлов топологии (сетевой, дорожный и станционный);
- разработана онтология для мониторинга и поддержки принятия решений при управлении движением железнодорожного транспорта;
- определены правила и реализованы средства ведения динамической объектной модели элементов инфраструктуры и процесса перевозок;
- реализованы средства и шаблоны формирования, утверждения, выполнения и мониторинга исполнения технологического процесса станции с разделением этих функций по уровням иерархии распределенной системы;
- подготовлены и реализованы шаблоны проектирования мнемосхем визуализации объектов инфраструктуры, подвижного состава, а также отображения режимов станций, переездов и ограничений движения;
- проведены подключения к внешним источникам данных по поездному местоположению, маршрутам и режимам станций, ремонтным "окнам" и предупреждениям; отработаны функции мониторинга доступности, управления записью, хранением и воспроизведением потока данных от внешних систем;
- реализованы компоненты имитационного программного обеспечения, эмулирующие состояние объектов инфраструктуры и объектов перевозочного процесса, а также внешних воздействий на анализируемый полигон железной дороги.

Интеграционная платформа в реализации текущего прототипа используется для отработки управленческих и технологических решений. Она является также подготовительным этапом перед созданием интеллектуальных систем управления производственным процессом на железнодорожном транспорте.

Авторы выражают благодарность Российскому Фонду Фундаментальных Исследований и Министерству науки и образования РФ, поддержавшим разработки по созданию интеллектуальных систем управления железнодорожным транспортом в реальном времени по гранту РФФИ № 11-07-13119-офи-м-2011-РЖД и государственному контракту Минобрнауки РФ № 07.514.11.4094

Список литературы

1. **Moffat J.** Complexity Theory and Network Centric Warfare. CCRP Publication Series. 2003. 201 p. URL: http://www.dodccrp.org/files/Moffat_Complexity.pdf

2. **Затуливер Ю. С.** Компьютерный базис сетецентрического управления // Труды Второй Российской конференции с международным участием "Технические и программные средства систем управления, контроля и измерения" (УКИ-10), Москва, 18—20 октября 2010 г. М.: ИПУ РАН, 2010. С. 492—511.

3. **Alford K., Ditmeyer S.** Network-Centric Operations: Defense and Transportation Synergy // CROSSTALK: The Journal of Defense Software Engineering. 2007. January. P. 20—23.

4. **Ditmeyer S.** Network-Centric Railways Operations Utilizing Intelligent Railway Systems // SCORT/TRB Rail Capacity Workshop. Jacksonville, Florida. 22 September 2010. URL: http://www.transportation.northwestern.edu/docs/2011/2011.03.15.Ditmeyer_Paper.pdf.

5. **Шабунин А. Б., Чехов А. В., Дмитриев Д. В., Курбатов Е. В., Сазуров С. В., Скобелев П. О., Симонова Е. В., Царев А. В., Степанов М. Е.** Сетецентрический подход к разработке системы управления производственными процессами ОАО "РЖД" // Труды международной научно-практической конференции "Управление большими системами—2011" (УБС'2011). Москва, 14—16 ноября 2011 г. М.: Изд-во ИПУ РАН, 2011. Т. 3. С. 222—225.

6. **Шабунин А. Б., Чехов А. В., Сазуров С. В., Курбатов Е. В., Дмитриев Д. В., Кузнецов Н. А., Скобелев П. О., Бабанин И. О., Кожевников С. С., Симонова Е. В., Степанов М. Е., Царев А. В.** Разработка сетецентрического подхода к созданию интеграционной платформы и интеллектуальной системы управления ресурсами грузовых сортировочных станций в реальном времени // Труды Третьей российской конференции с международным участием "Технические и программные средства систем управления, контроля и измерения" (УКИ'12). Москва, 16—19 апреля 2012 г. М.: Изд-во ИПУ РАН, 2012. С. 1560—1572.

7. **Шабунин А. Б., Чехов А. В., Скобелев П. О., Кузнецов Н. А., Симонова Е. В., Бабанин И. О., Кожевников С. С., Степанов М. Е., Царев А. В., Сазуров С. В., Курбатов Е. В., Дмитриев Д. В.** Сетецентрический подход к созданию распределенных систем управления ресурсами ОАО "РЖД" на основе мультиагентных технологий // Труды XIV Международной конференции "Проблемы управления и моделирования в сложных системах". Самара, 22—25 июня 2012 г. Самара: Изд-во СНЦ РАН, 2012. С. 724—734.

8. **Фаулер М.** Предметно-ориентированные языки программирования. СПб.: Вильямс, 2011.

УДК 004.82

И. С. Сысоев, аспирант,
Институт систем информатики им. А. П. Ершова, г. Новосибирск,
e-mail: jsysoev@yandex.ru

Ì îäåëü ðàññóæäåíèé íà îñíîâå ñòåðåîòèïîâ

Дана концепция подхода к представлению обыденного знания, который предположительно позволяет упростить данную задачу. Подход основан на символической модели образов прототипических ситуаций. Его ключевой особенностью является ориентированность на прагматику представляемого знания. Посредством этого исключается необходимость точного определения семантики понятий. Представлены как неформальное описание концепции, так и формальное определение простой вычислительной модели, реализующей данный подход.

Ключевые слова: *представление знания, обыденное знание, образное мышление, стереотип*

Введение

Задача представления обыденных знаний — т. е. знаний, которые человек сознательно и бессознательно использует в своей ежедневной жизни — является одной из интереснейших проблем в науке об искусственном интеллекте. К настоящему времени в некоторых парадигмах представления знаний были разработаны приемы представления самых разных видов обыденного знания[1]. Тем не менее задачу представления обыденного знания нельзя считать решенной. Это связано с большой трудоемкостью представления знаний в рамках традиционно используемых подходов, не позволяющей компьютерным системам приблизиться по объему знаний к человеку.

Отмеченная проблема, по мнению автора, порождена традиционным подходом к представлению знания через явное определение его семантики. Поскольку обыденные понятия не имеют строгой семантики, то использование их в формальной системе ведет к многочисленным несогласованностям (примеры см. в работе [2, разд. 6]). Таким образом, специалисты по представлению обыденного знания вынуждены формировать искусственные системы понятий, что создает значительные издержки. Во-первых, введение искусственной системы понятий требует кропотливого анализа представляемого фрагмента действительности, продолжительного проектирования и немалого архитектурного искусства. Во-вторых, разработчикам и пользователям интеллектуальной системы необходимо искать пути выражения обыденных явлений на искусственном языке, что зачастую весьма нетривиально. В-третьих, остается неясным, существует ли универсальная онтология верхнего уровня, которую можно адаптировать к любому знанию, которое хотелось бы представить [1, стр. 442]. На практике это означает, что очередной новый вид знания может потребовать переработки базовых понятий. Поскольку для обыденного знания характерны разнообразие его видов и изменчивость, сложно избежать этого риска. В результате, существующие базы обыденного знания можно сравнить с огромными монолитными сооружениями, которые требуют тщательного проектирования, трудоемки в возведении и проблематично перестраиваются.

В свете отмеченных выше издержек актуально изучение того, как рассуждение может базироваться на понятиях без формальной семантики (аналогично обыденной логике человека). Данный вопрос интересовал многих исследователей в различных дисциплинах, и точки зрения некоторых из них достаточно похожи. В философии интересна позиция Витгенштейна, выдвинувшего концепцию представления понятий через образы их прототипических экземпляров [3]. В лингвистике существует большое число работ по сходной тематике, хороший обзор которых можно найти в ра-

[1] С некоторыми приемами представления обыденного знания разного рода посредством логики предикатов, описательных логик и семантических сетей можно ознакомиться в работе [1, стр. 440—483].

боте [4]. В когнитивной психологии к данному направлению относится теория перцептуальных символов [5], в которой воспоминания, связанные с понятием, организуются в ментальный симулятор данного понятия. С позиций искусственного интеллекта существуют подходы Case-Based Reasoning [6] и Memory-Prediction Framework [7], в которых рассуждение основано на апелляции к запомненному образу аналогичного прошлого случая. Все эти исследования можно обобщить как *образно-ориентированный подход* к представлению знания. Насколько известно автору, как образно-ориентированный подход, так и задача представления обыденного знания редко затрагивались в отечественных публикациях.

Целью настоящей работы являлась оценка возможности создания метода представления знаний в виде образов, ориентированного на ввод знания человеком. Ручной ввод знаний, по мнению автора, является на данный момент актуальным. Это обусловлено отсутствием средств, способных предоставить количественно и качественно адекватный для обучения обыденному знанию объем информации[2], а также отсутствием должным образом развитых технологий формирования понятий в процессе обучения[3]. Кроме того, настоящий метод представления знаний полезен для апробации различных концепций образно-ориентированного подхода в целом, поскольку является удобным для восприятия человеком. Настоящая статья описывает результат первого этапа исследований на этом направлении, а именно — модель рассуждения на основе стереотипов (МРС). Единицей знания в данном подходе является стереотип — модель образа прототипической ситуации.

Данная работа в определенном смысле дополняет исследования в области формального представления понятий, описанные в работах [8, 9], где были получены важные результаты относительно формальных глоссариев — одного из способов явного представления понятий. В настоящей работе рассматривается подход к оперированию понятиями, лишенными явного определения. Одним из применений данного подхода является его совместное использование с механизмами, описанными в работе [9], в специализированных системах информационного поиска.

Идея МРС была высказана в работе [10]. Целью данной статьи является более подробное и точное определение модели.

1. Неформальное описание концепции

Рассмотрим концепцию представления знания в виде образов прототипических ситуаций. Обсудим специфику, преимущества и недостатки передачи знаний посредством образов. Рассмотрим вариант реализации рассуждений на основе стереотипов (формальное определение модели будет дано далее в разд. 2). Также сравним предложенную модель с другими вычислительными моделями образно-ориентированного мышления.

1.1. Образы

Будем исходить из предположения, что люди осуществляют коммуникацию посредством продуцирования образов в сознании собеседника. Предположим, что образы играют существенную роль также и в человеческом мышлении. Ярким примером образов, способных играть значительную роль в мышлении, являются стереотипы, распространенные в массовой культуре.

Первым шагом к моделированию образно-ориентированных процессов является введение модели образа. Ограничимся моделированием образов различных ситуаций (например, встречи двух друзей или строительства высотного здания), поскольку предполагаем, что все прочие образы могут быть разложены в систему образов ситуаций. Образ ситуации может задействовать разнообразные лица, существа, предметы, места, моменты времени и т. п., которые обобщенно будем называть *актерами*. В образ также входят взаимоотношения и взаимодействия различных актеров. Чтобы избежать путаницы с математическим понятием отношения, будем называть такие взаимодействия связями. Наша модель образа ситуации представляет собой набор актеров, объединенных связями. Актеры могут быть в большей или меньшей степени абстрактными, например, "красный автомобиль", а не конкретный автомобиль с номером "А 574 НБ". Более или менее абстрактными могут быть и связи, например, "родственники" и "троюродные братья".

Будем называть *стереотипом* образ ситуации, использующийся в процессе рассуждения как модель, посредством которой интерпретируются наблюдаемые факты. Данное определение соответствует обыденному пониманию стереотипа. Уже упоминавшиеся стереотипы в массовой культуре могут быть представлены посредством одного или нескольких стереотипов в терминологии настоящего подхода.

Процесс передачи знаний, основанный на образах, обладает достаточно большой спецификой. В отличие от утверждения, образ не имеет определенной семантики. Если говорящему и удается передать определенное сообщение слушателю, то это связано не с пониманием обоими точной семантики сообщения, а с тем обстоятельством, что они имеют сходные картины мира. В рамках этих сходных картин одно предложение вызывает сходные образы, которые сходно функционируют в мышлении. Таким образом, можно утверждать, что интересующий нас процесс базируется не на семантике, а на прагматике сообщения.

С одной стороны, подобный способ передачи сообщений отличается большой вероятностью неверной интерпретации. Впрочем, некоторые особенности процесса коммуникации между людьми частично ком-

[2] Аргументацию в пользу того, что ручной ввод обыденного знания является на данный момент необходимым, можно найти в работе [10].

[3] Одна из существующих технологий упоминается в работе [1, стр. 937—939].

пенсируют этот недостаток. Во-первых, обычно имеет место следование подряд идущих предложений общей теме, что позволяет отбрасывать интерпретации, в которых предложение резко контрастирует со своими соседями. Во-вторых, надежности способствует интерактивный характер процесса общения. Действительно, слушатель имеет возможность уточнять смысл получаемых сообщений, а говорящий — проверять, правильно ли они были поняты. Другую разновидность обратной связи процесса получения знаний можно наблюдать у детей. Часто можно видеть, как они рассказывают взрослым придуманные истории про тех людей и предметы, которые они недавно видели или про которые слышали. Подобные языковые игры, вероятно, позволяют им осваивать новое знание и корректировать его на основе наблюдения за реакцией взрослых.

С другой стороны, именно отсутствие у образов строгой семантики делает их столь удобным инструментом представления и передачи обыденного знания. Во-первых, образы позволяют описывать понятия, с трудом поддающиеся формализации, путем простого перечисления типовых случаев их использования. Как отмечается в работе [8], трудность формализации характерна для обыденных понятий, поэтому данное свойство является весьма востребованным. Заметим, что постановка проблемы описания понятий и рассматриваемый вариант ее решения не новы, они были проанализированы еще в трудах Л. Витгенштейна [3].

Во-вторых, в то время как строгие утверждения являются общезначимыми, каждый образ ассоциируется только с узким кругом ситуаций. Причем круг может автоматически сужаться, когда к базе знаний добавляются образы, представляющие частные случаи из данного круга. В результате достигается локальность вносимых изменений, а также гибкость добавления нового знания. Из широко известных подходов свойством автоматического определения области действия фрагмента знания обладает логика косвенного описания (см., например, [12, 13]). Хотя логика косвенного описания требует явной спецификации правил по умолчанию и исключений из них, были проведены исследования по автоматизации этого процесса [14]. Тем не менее логика косвенного описания не имеет других перечисляемых здесь преимуществ образно-ориентированного подхода.

В третьих, образы могут выступать в качестве звеньев ассоциативной цепи, связывающих различные способы представления одинаковых фактов. По-видимому, именно эта особенность позволяет людям общаться без необходимости предварительного ознакомления с архитектурой представления знания в сознании друг друга. В искусственных системах данная особенность позволила бы интегрировать разнородные базы знания, в том числе созданные для различных предметных областей. Задача интеграции знания, извлеченного из обыденного текста, возникает также в вопросно-ответных и аналитических системах [15]. По этой причине можно ожидать, что для данных областей образно-ориентированный подход также окажется полезным.

В-четвертых, образ может быть более гибко вовлечен в рассуждение, нежели традиционные правила "если-то", в современной практике являющиеся основной формой записи знания. Для примера представим образ концерта симфонического оркестра. Прежде всего, этот образ позволяет понять что перед нами, когда мы видим множество людей во фраках со смычковыми и духовыми инструментами. Зная, что перед нами оркестр, мы можем установить, что человек, размахивающий палочкой — это дирижер. Если нас спросят, что находится в правой руке каждого скрипача, мы можем ответить: смычок. Если же вместо этого нас попросят указать, где находятся смычки, то мы ответим, что в руках скрипачей. Таким образом, в зависимости от возникающих потребностей, один и тот же образ можно использовать для огромного числа различных заключений, заменяя целый набор правил "если-то". Способность к гибкому применению знания, по-видимому, является важной составляющей интеллектуального поведения.

В контексте формального представления знаний приведенная выше модель образа может обладать еще одним преимуществом. Если наше предположение о природе коммуникации между людьми корректно, то данный способ представления знаний должен являться наиболее естественным для разработчика интеллектуальной системы. Это обстоятельство должно способствовать эффективности его работы.

1.2. Процесс рассуждения

Выше была введена модель образов. Рассмотрим, как на основе этой модели может быть построен процесс рассуждения.

Прежде всего необходимо установить, какого рода выводы являются востребованными высокоуровневыми приложениями. Представляется, что высоко востребованы три перечисленные далее пары исходных данных и запроса.

1. Пусть задана некоторая неполная ситуация; какие не упомянутые в исходной ситуации факты правдоподобны?

2. Пусть задана некоторая ситуация и некоторый факт; какими путями ситуация может быть расширена, чтобы объяснить данный факт?

3. Пусть задана некоторая ситуация и некоторый факт; является ли данный факт неправдоподобным в данной ситуации?

Запросы первого типа позволяют системе строить картину мира, исходя из неполных данных, прогнозировать развитие событий и последствия ее собственных действий. Запросы второго типа позволяют выделять возможные причины явлений в целях определения перспективных направлений анализа ситуации, а также находить действия, которые приводят к определенной цели. Запросы третьего типа требуются прежде всего в задачах, касающихся естественного языка, для отсеивания маловероятных побочных значений предложения.

В каждом из указанных трех случаев вывод может быть получен по одному общему принципу. Ситуацию, заданную в запросе, назовем *фокусной ситуацией*. В распоряжении машины вывода есть набор стереотипов, представленных на необходимо высоком уровне абстракции. Такая машина выводит заключения путем нахождения стереотипов, отождествимых с различными фрагментами фокусной ситуации, и переноса их характеристик на отождествленные фрагменты.

В настоящей работе сосредоточимся на запросах первого типа. Для данных запросов определяем следующую процедуру обработки.

1. Нахождение стереотипов, имеющих элементы, сходные (таксономически и по связям друг с другом) с элементами фокусной ситуации. Может существовать несколько элементов фокусной ситуации, подобных некоему элементу стереотипа. Возможные варианты отображения элементов стереотипа на сходные элементы фокусной ситуации будем называть *отождествлениями*. На следующий этап поступают только те отождествления, при которых не возникает противоречия между свойствами соответствующих элементов.

2. Проверка, не подавляются ли некоторые отождествления очевидно лучшими отождествлениями — например, когда перестановка отождествленных элементов выявляет более точное соответствие стереотипа и фокусной ситуации. В каждом таком случае "худшее" отождествление отбрасывается. Было разработано несколько правил, по которым осуществляется выбор предпочтительных отождествлений. Заметим, что по причине, которая будет указана в подразд. 1.3, было принято решение воздержаться от использования метрик в этих правилах.

3. При помощи каждого из оставшихся отождествлений расширение фокусной ситуации, с помощью переноса на нее фактов, имеющих место в отождествленных стереотипах. Эта процедура называется *применением отождествлений*.

4. Продолжение процесса с новыми отождествлениями, которые стали возможны после применения фокусной ситуации. Определенный таким образом процесс рассуждения является одной из форм прямого логического вывода.

1.3. Родственные модели

Среди большого числа различных моделей осуществления рассуждений есть несколько таких, которые явно или неявно задействуют образы. Остановимся подробнее на двух моделях, которые можно считать до определенной степени близкими МРС.

Первой из них является модель "память-предсказание" (*Memory-Prediction Framework*) [7]. Она была предложена как теория функционирования коры головного мозга (объясняющая некоторые особенности ее строения) и одновременно как принцип, который может быть использован при построении универсальных интеллектуальных машин. Данная теория основана на предположении, что разум может быть представлен как иерархическая система памяти. При этом в основе процесса мышления на каждом уровне иерархии лежит сопоставление поступающих сигналов с заученными паттернами и передача сигнала о соответствии на следующий уровень иерархии. Одновременно на низшие уровни иерархии передаются извлеченные из паттернов текущего уровня подсказки, какие образы низших уровней должны восприниматься в данный момент. Эти подсказки помогают в разрешении неоднозначностей в зашумленных данных, в восстановлении отсутствующих фрагментов входного сигнала (частично закрытое изображение), в фокусировке внимания и т. д. Модель "память-предсказание" является попыткой низкоуровневого описания работы мозга, поэтому в качестве образа в данной модели выступает иерархия пространственно-временных паттернов активации нейронов. В связи с этим обстоятельством, несмотря на сходство базовых идей, родство модели "память-предсказание" с МРС является отдаленным.

В отличие от первой модели, модель рассуждений на основе прошлых случаев (*Case-Based Reasoning*, CBR, см., в частности, [6]) оперирует высокоуровневыми концепциями. Изначально данная модель была предложена для осуществления рассуждений общего вида и использования в программах автоматического перевода [16]. В настоящее время этот подход чаще всего рассматривается в приложении к экспертным системам для решения различных задач. Поиск решения задачи проводится путем поиска наиболее близкого из прошлого опыта случая и адаптации использовавшегося в прошлом случае решения к текущей ситуации. Модель CBR имеет множество достаточно сильно отличающихся друг от друга вариаций, как, например, модель рассуждений на основе экземпляров, модель рассуждений на основе аналогий и т. д. [6]. Фактически МРС также можно считать одной из вариаций CBR. Однако в то время как CBR обычно рассматривается как нетрадиционный подход к машинному обучению, МРС ориентирована на ручной ввод знаний. Это важное отличие МРС от типовых реализаций CBR влечет за собой несколько других.

Первое отличие заключается в том, что стереотипы не описывают конкретную ситуацию. Вместо этого они представляют собой обобщенные, весьма абстрактные описания целого класса конкретных ситуаций. Так, например, "короли носят корону" является стереотипом, в то время как "король Георг V носил корону Британской Империи" — это конкретная ситуация. Представляется, что ввод одной абстрактной ситуации значительно удобнее, чем ввод множества примеров, иллюстрирующих эту ситуацию.

Для того чтобы быть в достаточной степени абстрактными, стереотипы не должны включать большое число деталей. Однако это ограничение означает, что в процессе рассуждения необходимо отождествлять фокусную ситуацию с большим числом стереотипов вместо того чтобы проводить аналогии с одной конкретной ситуацией. Более того, данный процесс, по всей видимости, должен быть итеративным. Это делает процесс вывода в МРС более напоминающим типовой

процесс вывода в продукционных системах, нежели типовой вывод в парадигме CBR.

Наконец, в MPC не используется метрика. Вместо этого применяется сравнение так называемых проекций фокусной ситуации на стереотип при помощи специально определенного отношения обобщения ситуаций. Проекция представляет ту часть фокусной ситуации, которая соответствует стереотипу. Такое решение было выбрано в связи с тем, что использование метрик имеет тенденцию приводить к появлению множества числовых коэффициентов, распределенных по всей базе знаний. В больших базах знаний чрезвычайно сложно поддерживать согласованность этих коэффициентов. Это обстоятельство является причиной, по которой в Cyc, крупнейшем на настоящее время проекте по представлению обыденного знания, не используются коэффициенты уверенности [11]. Вместо этого для разрешения конфликтов вывода применяется сопоставление аргументов в пользу каждого из противоречащих суждений [17]. Применяемый в MPC подход аналогичен этому решению.

2. Формальное определение модели

Дадим формальное определение модели рассуждений, описанной в подразд. 1.2, а также определение необходимых понятий и механизма рассуждения.

2.1. Ситуации

Начнем с определения модели образа ситуации. Для краткости далее будем именовать это понятие просто *ситуация*. Ситуация может играть роль фокусной ситуации либо стереотипа.

Как уже отмечалось выше, ситуация задействует актеров и связи между ними. Каждый актер описывается при помощи некоторого множества *характеристик*. Каждой связи ставится в соответствие некий *тип связи*.

Характеристика — это некий идентификатор, который используется для обозначения либо свойства актера (например, красный), либо его таксономической принадлежности (например, птица). Уникальные характеристики используются для обозначения конкретных объектов (например, Москва). Множество характеристик актера a в контексте ситуации s обозначается как $features_s(a)$.

Тип связи $type(c)$ также является неким идентификатором. Каждому типу связи поставлено в соответствие множество идентификаторов, называемых позициями: $P(type(c))$. Каждой позиции $p \in P(type(c))$ связи c поставлен в соответствие некий актер, обозначаемый $c[p]$ и называемый аргументом связи.

На множестве характеристик заданы частичный порядок $\mathtt{m}_F$ и отношение $<>_F$. Отношение $\mathtt{m}_F$ указывает, что правая характеристика является более общей, например, *собака* $\mathtt{m}_F$ *млекопитающее*. Отношение $<>_F$ указывает несовместные пары характеристик, например, *собака* $<>_F$ *кошка*. Аналогично этому на множестве типов связей заданы частичный порядок степени общности $\mathtt{m}_{CT}$ и отношение несовместности $<>_{CT}$. Например, *братья* $\mathtt{m}_{CT}$ *родственники*, *легче* $<>_{CT}$ *тяжелее*. Отношения типов связей определяют соответствующие отношения между связями данных типов, имеющими одинаковые аргументы в общих позициях. Отношения между связями будут определены в подразд. 2.2.

Ситуация s представляется парой $(A(s), C(s))$, где $A(s)$ — множество актеров и $C(s)$ — множество связей.

Ниже дан пример ситуации в одной из возможных нотаций. Он соответствует представлению "короли носят на голове корону":

$$c_1 : часть(что = a_1 : \{голова\},\ чего = a_2\{король\})$$

$$c_2 : расположен - на(что = a_3 : \{корона\},\ чём = a_1),$$

где: $A(s) = \{a_1, a_2, a_3\}$; $C(s) = \{c_1, c_2\}$; $features(a_3) = \{корона\}$; $type(c_1) = часть$; $S(type(c_1)) = \{что, чего\}$; $c_1[что] = a_1$.

Для фокусной ситуации fs истинные утверждения вида "актер a имеет характеристику f в контексте fs" и "связь c принадлежит $C(fs)$" назовем фактами.

Данное определение ситуации можно назвать статическим, поскольку оно не отражает развитие ситуации во времени. Принятие во внимание временного аспекта является важным направлением дальнейшей работы.

2.2. Сопоставление ситуаций

Для того чтобы рассуждение было возможным, необходимо как-то сопоставлять ситуации друг с другом. Определенный выше процесс рассуждения интенсивно использует два отношения между ситуациями: отношение обобщения и отношение отождествимости фокусной ситуации и стереотипа. В качестве вспомогательного также рассмотрим отношение эквивалентности ситуаций. Каждое из этих отношений связано с определенным осмысленным отображением актеров и связей из одной ситуации в другую.

Пусть $am : A(s_1) \to A(s_2)$ есть некоторое отображение актеров одной ситуации в актеров другой. Связи c из s_1 поставим в соответствие так называемый *am-образ* данной связи. Он является новой связью с тем же типом, что и c, а аргумент *am*-образа c в позиции p по определению равен $am(c[p])$.

Ситуация s_1 и ситуация s_2 называются эквивалентными тогда и только тогда, когда существует биективное отображение $am : A(s_1) \to A(s_2)$ такое, что множество характеристик актера a из $A(s_1)$ совпадает со множеством характеристик $am(a)$ и множество связей s_2 совпадает со множеством *am*-образов всех связей из $S(s_1)$. Данное отношение обозначается $\sim_s$.

Утверждение 1. Отношение $\sim_s$ действительно является эквивалентностью.

Две связи называются согласованными по аргументам тогда и только тогда, когда они имеют одинаковых актеров в каждой общей позиции (т. е. $\forall p \in P(type(c_1)) \cap \cap P(type(c_2))\ (c_1[p] = c_2[p])$).

Пусть R — некоторое отношение на связях. Полагаем, что связь c_1 находится в отношении R со связью c_2 при am, где am — некоторое отображение актеров,

тогда и только тогда, когда *am*-образ c_1 находится в отношении R с c_2.

Считаем, что актер a_1 ситуации s_1 является более общим, чем актер a_2 ситуации s_2, тогда и только тогда, когда каждая характеристика f_1 из $features_{s1}(a_1)$ является более общей, чем некоторая характеристика f_2 из $features_{s2}(a_2)$ ($f_2 \, \mathtt{m}_F f_1$). Полагаем, что связь c_1 является более общей, чем связь c_2 тогда и только тогда, когда тип c_1 является более общим, чем тип c_2, и связи согласованы по аргументам. Отношение «c_1 является более общей связью, чем c_2, при отображении актеров *am*» определяется по указанному выше шаблону.

Теперь можем определить отношение обобщения на ситуациях. Пусть s_1 и s_2 — некоторые ситуации. Полагаем, что s_1 *является более общей, чем* s_2 *при am* (где *am* — инъективное отображение из $A(s_1)$ в $A(s_2)$) тогда и только тогда, когда всякий a из $A(s_1)$ является более общим, чем $am(a)$, и всякая связь c_1 из $C(s_1)$ является более общей, чем некоторая связь c_2 из $C(s_2)$ при *am*. Считаем, что s_1 *является более общей, чем* s_2, тогда и только тогда, когда существует отображение *am*, такое что s_1 является более общей, чем s_2 при *am*. Данные отношения обозначаются соответственно $\mathtt{m}_S^{am}$ и $\mathtt{m}_S$.

Отношение обобщения на ситуациях обладает интересными свойствами, если рассматриваются так называемые *лаконичные* ситуации. Ситуация s называется лаконичной, если она не содержит двух различных связей $c_1 \in C(s)$ и $c_2 \in C(s)$, таких, что c_1 является более общей, чем c_2, и для всякого $a \in A(s)$ не существует различных характеристик f_1 и f_2 из $features_s(a)$ таких, что $f_1 \, \mathtt{m}_F f_2$.

Всякую ситуацию s можно привести к лаконичному виду путем исключения более общих характеристик и связей из каждой пары, нарушающей требование лаконичности. Полученная ситуация называется *лаконичной версией s*. Лаконичная версия ситуации описывает то же положение дел, что и исходная ситуация. Из соображений удобства, а также из соображений экономии памяти при реализации на ЭВМ, требуем лаконичности всех ситуаций, с которыми работаем.

Для лаконичных ситуаций верны перечисленные далее свойства отношения общности ситуаций.

Утверждение 2. Для любого инъективного отображения актеров *am* отношение $\mathtt{m}_S^{am}$ является частичным порядком.

Утверждение 3. Отношение $\mathtt{m}_S$ рефлексивно и транзитивно.

Утверждение 4. Ситуации s_1 и s_2 взаимно находятся в отношении $\mathtt{m}_S$ тогда и только тогда, когда они эквивалентны.

Данные свойства отношений общности позволят использовать их в правилах предпочтения, которые определяются в следующем подразд.

Далее определим отношение отождествимости фокусной ситуации и стереотипа.

Во-первых, считаем, что актер a_1 ситуации s_1 отождествим с актером a_2 ситуации s_2 тогда и только тогда, когда не существует характеристик $f_1 \in features_{s_1}(a_1)$ и $f_2 \in features_{s_2}(a_2)$, которые являются несовместными ($f_1 <>_F f_2$).

Связь c_1 является отождествимой со связью c_2 тогда и только тогда, когда они согласованы по аргументам и имеют *сравнимые* типы. Последнее означает, что имеет место либо $type(c_1) \, \mathtt{m}_{CT} \, type(c_2)$, либо $type(c_2) \, \mathtt{m}_{CT} \, type(c_1)$.

Считаем, что связь c_1 несовместна со связью c_2 тогда и только тогда, когда связи согласованы по аргументам и обладают несовместными типами ($type(c_1) <>_{CT} type(c_2)$).

В соответствии с указанным ранее шаблоном определяются отношения отождествимости и несовместности связей при отображении актеров *am*.

Рассмотрим множество актеров A, множество связей C и граф, ребра которого соединяют все связи из C и их аргументы из A. A называется *соединенным посредством* C тогда и только тогда, когда данный граф является связным.

Пусть s — стереотип, а fs — фокусная ситуация. Пусть *am* — инъективное отображение актеров из некоторого непустого подмножества $A(s)$ в $A(fs)$, а *cm* — отображение связей из некоторого подмножества $C(s)$ в $C(fs)$. Пусть отображения обладают следующими свойствами:

- всякий актер a из области определения *am* является отождествимым с $am(a)$;
- всякая связь c из области определения *cm* является отождествимой с $cm(c)$;
- не существует связей c_s из $C(s)$ и c_{fs} из $C(fs)$, являющихся несовместными при *am*;
- образ $A(s)$ при отображении *am* соединен посредством образа $C(s)$ при отображении *cm*.

В этом случае пара (*am*, *cm*) называется отождествлением s с фокусной ситуацией fs.

Последнее условие в определении отождествления позволяет предотвратить домысливание связей между не имеющими отношения друг к другу объектами фокусной ситуации.

Стереотип s называется отождествимым с фокусной ситуацией fs, если существует отождествление s с fs.

2.3. Контроль отождествления

Стереотип является отождествимым с фокусной ситуацией в том случае, если присутствует некоторое соответствие (возможно, слабое) их содержимого. Однако определение отождествимости не учитывает, что одни отождествления выявляют лучшее соответствие стереотипа и фокусной ситуации нежели другие. За установление этого факта ответственны эвристические *правила предпочтения отождествлений*.

Правило лучшей гипотезы является примером правила предпочтения. Данное правило задействует два определения: *существенной части* стереотипа и *проекции* фокусной ситуации на стереотип.

Существенная часть стереотипа задается инженером знания с целью указать, что стереотип посвящен описанию какого-то феномена. Для этого инженер знания помечает как существенные некоторые характеристики некоторых актеров стереотипа и некоторые

из его связей. Например, если стереотип посвящен описанию канареек, то должна быть помечена как существенная характеристика *канарейка* одного из его актеров. *Существенная часть* стереотипа определяется как ситуация, которая включает все существенные связи стереотипа, все их аргументы и всех актеров, обладающих существенными характеристиками. Множество характеристик актера в контексте существенной части равно множеству его существенных характеристик в исходном стереотипе. Существенная часть s обозначается $essential(s)$. Заметим, что существенная часть лаконичного стереотипа также будет лаконичной.

Проекция фокусной ситуации на стереотип при некотором их отождествлении определяется как часть фокусной ситуации, которая соответствует стереотипу. Пусть s — стереотип, fs — фокусная ситуация, (am, cm) — некоторое их отождествление. Проекция является лаконичной версией определяемой далее ситуации pr. Множество актеров pr равно области значений am. Для каждого актера a из $A(pr)$ и соответствующего ему актера $a_0 = am^{-1}(a)$ определяем $features_{pr}(a)$ как множество более общих характеристик по всем сравнимым парам характеристик a и a_0[4]. Множество $C(pr)$ состоит из более общих связей по всем парам $(cm(c), am - образ(c))$, где c — некая связь из $C(s)$, принадлежащая области определения cm. Проекция fs на s при отождествлении i обозначается как $proj_i(fs, s)$.

Идея правила лучшей гипотезы заключается в следующем. Отождествление стереотипа и фокусной ситуации в определенном смысле является эмпирической гипотезой, объясняющей некоторые факты из фокусной ситуации. Если одно отождествление объясняет более широкий набор фактов, нежели второе, то первому должно быть отдано предпочтение, поскольку оно является лучшей гипотезой состояния вещей в фокусной ситуации. Если два отождествления объясняют одинаковое множество фактов, то должны быть отброшены оба, поскольку нет возможности сделать выбор между гипотезами. Исключением является случай, когда отбрасываемое отождествление отображает существенную часть стереотипа на фокусную ситуацию. При таком отождествлении стереотип, несомненно, имеет отношение к фокусной ситуации, поскольку он *посвящен* описанию того, что в фокусной ситуации имеет место. В этом случае отождествление сохраняется.

Ниже приводится формальное определение данного правила. Пусть s_1 и s_2 — два стереотипа, $i_1 = (am_1, cm_1)$ и $i_2 = (am_2, cm_2)$ — их отождествления с фокусной ситуацией fs соответственно, а pr_1 и pr_2 — соответствующие проекции. Если $pr_1 \neq pr_2$ и pr_2 является более общей, чем pr_1 при тождественном отождествлении актеров, и $essential(s_2)$ не является более общей, чем pr_2 при am_2, то i_2 должно быть отброшено. Если $pr_1 = pr_2$, то должны быть отброшены те отождествления, существенная часть соответствующего стереотипа которых не является более общей, чем соответствующая проекция при соответствующем отображении актеров.

С использованием сравнения общности проекций могут быть сконструированы некоторые другие интересные правила предпочтения. В настоящее время ведется апробация этих правил.

2.4. Применение отождествлений

Конечной целью нахождения отождествлений и их последующей фильтрации по правилам предпочтения является расширение фокусной ситуации фактами, которые могут быть получены из предпочтительных отождествлений. Операция расширения фокусной ситуации при помощи подходящего отождествления называется *применением отождествления*. Результат применения к фокусной ситуации fs отожествления i стереотипа s с той же фокусной ситуацией обозначается $fs + i$. Он является лаконичной версией ситуации $(fs + i)_0$, которая определяется следующим образом.

Пусть $i = (am, cm)$. $(fs + i)_0$ содержит всех актеров фокусной ситуации, а также новых актеров, соответствующих неотождествленным актерам стереотипа. Если a_{fs} из $A(fs)$ не принадлежит области значений am, то он сохраняет свой старый набор характеристик в $(fs + i)_0$. Если a_{fs} из $A(fs)$ является образом некоего a_s из $A(s)$ при отображении am, то множество характеристик a_{fs} в $(fs + i)_0$ состоит из характеристик обоих актеров: $features_{(fs+i)_0}(a_{fs}) = features_s(a_s) \cup features_{fs}(a_{fs})$. Наконец, если a_s — это актер из $A(s)$, не принадлежащий области определения am, то в $(fs + i)_0$ ему будет соответствовать новый актер a_{new} со множеством характеристик, равным $features_s(a_s)$. am' — это am, дополненное отображением актеров последнего типа на новых актеров. Множество связей $(fs + i)_0$ является объединением множества связей fs со множеством am'-отображений всех связей из s.

2.5. Процесс рассуждения

Объединим данные выше понятия в формальном определении процесса рассуждения, который ранее был описан в подразд. 1.2.

Пусть дана фокусная ситуация fs_0, которую требуется расширить фактами, правдоподобными в ее контексте. Машина вывода имеет в своем распоряжении базу знаний $(F, CT, \mathtt{m}_F, \mathtt{m}_{CT}, <>_F, <>_{CT}, S)$, где F — множество характеристик, CT — множество типов связей, S — множество стереотипов. Процесс вывода происходит через повторение одинаковых шагов. Очередной k-й шаг осуществляет новое расширение фокусной ситуации, которое обозначается fs_k. Определим k-й шаг рассуждения следующим образом:

- I — это множество всех отождествлений стереотипов из S с fs_{k-1};
- $topI$ — это результат фильтрации I правилами предпочтения отождествлений;
- $fs_k = fs_{k-1} + i_1 + ... + i_n$, где $\{i_1 ... i_n\} = topI$.

[4] $features_{pr}(a) = \{f_1 | f_1 \in features_{fs}(a), f_2 \in features_s(a_0), f_2 \,\mathtt{m}\, f_1\} \cup \cup f_2 | f_1 \in features_{fs}(a), f_2 \in features_s(a_0), f_1 \,\mathtt{m}\, f_2\}$.

Ситуации $\{fs_k | k > 0\}$ называются *частичными результатами* рассуждения. Если на некотором шаге k получаем $fs_k = fs_{k+1}$, то fs_k называется *окончательным результатом* рассуждения. Может быть задана эвристика завершения процесса рассуждения до получения окончательного результата.

3. Перспективные направления работ

Следующим шагом исследования представляется разработка иерархической модели стереотипа. Прообразом иерархической модели является текущая двухуровневая иерархия "стереотип — существенная часть". Более развитые иерархии могут включать различные вариации и опциональные расширения стереотипа. Использование иерархической модели позволит более гибко вовлекать стереотипы в рассуждение. Однако основным преимуществом данной модели является возможность внедрения иерархии контекстов, аналогичных описанным в работе [10].

В более отдаленной перспективе должны быть предприняты шаги по устранению недостатков текущей модели. Одним из них является отсутствие поддержки аспекта времени. Другой недостаток — наличие иерархий общности, которые до определенной степени требуют явной концептуализации.

Заключение

Существует множество подходов к формальному представлению знаний. Большинство из них основано на определении формальной семантики той или иной формы записи утверждений. Это приводит к ряду затруднений при представлении обыденного знания. В настоящей работе представлен подход, сфокусированный не на семантике, а на прагматике вводимого знания. Настоящий подход исходит из предположения об образной природе повседневного мышления человека. В работе рассмотрена специфика, потенциальные недостатки и потенциальные преимущества образно-ориентированного способа представления знаний. Также была представлена простая вычислительная модель рассуждений, реализующая данный подход.

В заключение автор хотел бы поблагодарить своего научного руководителя Дмитрия Евгеньевича Пальчунова за ценные замечания, высказанные в ходе подготовки данной статьи.

Работа поддержана грантом Федеральной целевой программы "Научные и научно-педагогические кадры инновационной России" на 2009—2013 годы, проект "Методы извлечения и порождения знаний для обеспечения информационной безопасности".

Список литературы

1. **Рассел С., Норвиг П.** Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. М.: Вильямс, 2006.

2. **Woods W.** What is in a link: Foundations For Semantic Networks // Representation and Understanding: Studies in Cognitive Science / Ed. D. Bobrov, A. Collins. New York: Academic Press, 1975.

3. **Витгенштейн Л.** Философские исследования. Пер. с нем. А. Добросельского. М.: АСТ: Астрель, 2011. С. 58—65.

4. **Лакофф Д.** Женщины, огонь и опасные вещи: Пер. с англ. И. Шатуновского. М.: Языки славянской культуры, 2004.

5. **Barsalou L.** Perceptual Symbol Systems // Behavioral and Brain Sciences. 1999. № 22. P. 557—660.

6. **Aadmont A., Plaza E.** Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches // Artificial Intelligence Communications, 1994. Vol. 7. № 1. P. 39—52.

7. **Hawkins J., Blakeslee S.** On Intelligence. Times Books, Henry Holt and Co., 2005.

8. **Пальчунов Д. Е.** Моделирование мышления и формализация рефлексии. Ч. II. Онтологии и формализация понятий // Философия науки. 2008. № 2 (37). С. 62—96.

9. **Palchunov D. E.** Virtual catalog: the ontology-based technology for information retrieval. Lecture notes in computer science 6581. Springer-Verlag, 2011. P. 164—183.

10. **Сысоев И. С.** Представление обыденного знания в форме шаблонных ситуаций // Ершовская конференция по информатике 2011, тр. третьего семинара "Знания и онтологии *ELSEWHERE* 2011". Новосибирск: Институт систем информатики им. А. П. Ершова СО РАН, 2011. С. 96—102.

11. **Lenat D.** Cyc: A large-scale investment in knowledge infrastructure // Communications of ACM. 1995. Vol. 38. № 11. P. 32—38.

12. **McCarthy J.** Application of Circumscription to Formalizing Common-Sense Knowledge // Artificial Intelligence. 1986. № 28. P. 89—116.

13. **McCarthy J.** Circumscription — A Form of Non-Monotonic Reasoning // Artificial Intelligence. 1980. № 13. P. 27—39.

14. **Delgrande J., Schaub T.** Incorporating specificity into circumscriptive theories // Advances in Artificial Intelligence. Proceedings of the Eighteens German Annual Conference on Artificial Intelligence. Springer-Verlag, 1994. P. 272—282.

15. **Пальчунов Д. Е.** Поиск и извлечение знаний: порождение новых знаний на основе анализа текстов естественного языка // Философия науки. 2009. № 4 (43). С. 70—89.

16. **Schank R.** Dynamic Memory: A Theory of Learning in Computers and People. New York: Cambridge University Press, 1982.

17. **Guha R.** The representation of defaults in Cyc // Proceedings of AAAI-90. AAAI Press, 1990. P. 608—614.

УДК 004

В. В. Липаев, д-р техн. наук, проф., гл. науч. сотр.,
Институт системного программирования РАН,
e-mail vlip28@mail.ru

Èç èñòîðèè ñîçäàíèÿ îòå÷åñòâåííûõ ïðîãðàìì ðåàëüíîãî âðåìåíè è êîìïüþòåðíûõ ñåòåé äëÿ ñèñòåì ïðîòèâîâîçäóøíîé îáîðîíû

Рассматриваются основные задачи, которые решались на начальном этапе создания компьютерных сетей с вычислительными узлами для обработки в реальном времени радиолокационной информации о движущихся объектах противовоздушной обороны. Изложены возникавшие в связи с этим трудности и принятая в 1960-е гг. концепция построения программного обеспечения операционных систем реального времени для управления процессами обработки информации. Представлена архитектура комплекса программ для обработки и передачи радиолокационной информации о воздушных объектах в глобальной сети противовоздушной обороны.

Ключевые слова: *операционная система реального времени, телекоммуникационные сети, радиолокационные узлы, имитация воздушной обстановки*

Введение

Повышенное внимание к современным методам и средствам разработки и сопровождения сложных, многофункциональных программ в сочетании с острой нехваткой специалистов в этой области заставляют по-новому взглянуть на отечественный опыт прошлых лет с тем, чтобы оценить его с позиций сегодняшнего дня. В статье рассматриваются основные задачи, которые решались на начальном этапе истории создания компьютерных сетей с вычислительными узлами для обработки радиолокационной информации о динамических объектах (самолетах, ракетах, космических аппаратах) противоздушной обороны (ПВО) в реальном времени.

Высокие темпы развития средств вычислительной техники и коммуникационного оборудования за последние полвека превратили распределенные компьютерные системы различного назначения в современные, очень сложные аппаратно-программные инфраструктуры. В них используются сетевые компоненты, связывающие большое число узлов, предоставляющих пользователям вычислительные ресурсы для решения разных задач. Компьютерная сеть в настоящее время обеспечивает надежный, устойчивый и недорогой доступ пользователей к вычислительным ресурсам и различным данным. Создание и развитие компьютерных сетей в Советском Союзе начиналось в пятидесятые годы прошлого века в интересах оборонно-промышленного комплекса страны при полном отсутствии информации об аналогичных работах за рубежом.

Очень ограниченные ресурсы отечественных компьютеров **середины 1960-х годов** и их индивидуальное использование преимущественно для решения наиболее актуальных вычислительных задач того времени определяли "ручной" режим распределения администраторами времени, которое они считали возможным выделить программистам и/или пользователям программ. Потребности в компьютерном времени заставили автоматизировать процедуры регистрации и упорядочивания запросов специалистов на решение их задач. Это обстоятельство привело к созданию автоматизированных операционных систем распределения времени функционирования компьютеров с учетом индивидуальных потребностей и очередей пользователей. Такие системы ориентировались на потребности

в использовании компьютеров не взаимодействующими друг с другом специалистами в научных учреждениях и вузах при относительно небольшом спросе на их применение и небольших по размеру программах. Возрастание очередей на решение различных задач при ограниченном числе доступных потенциальным пользователям компьютеров привело к необходимости совершенствования механизмов управления использованием ресурсов отдельных вычислительных машин.

Результаты первых поисковых исследований и практических работ в **1957 г.** в НИИ-5 Министерства обороны СССР послужили прототипами для создания ***сложной глобальной телекоммуникационной сети радиолокационных узлов (РЛУ)***. В рамках этих работ моделировались исходные координаты, отражающие состояние движения наблюдаемых радиолокаторами объектов, которые могли составлять группы и маневрировать. По этим данным имитировалось обнаружение воздушных объектов, формировались траектории их движения в пространстве. Отождествлялись и объединялись расчетные траектории динамических объектов в реальном времени от различных РЛУ. В **1959 г.** руководителями и специалистами этого института был проведен анализ и обобщение выполненных исследований. Их результаты нашли отражение в отчете по большой комплексной работе, направленной на создание территориально-распределенной информационной системы ПВО страны — проект "Электрон". Эта система должна была впервые в отечественной практике объединить компьютеры, работающие в глобальной сети, в режиме реального времени на узлах сбора и обработки радиолокационной информации о воздушных объектах, а также на командных пунктах управления активными средствами ПВО.

Все средства обработки информации и управления в системе должны были работать на ЭВМ в реальном времени, при несинхронных потоках сообщений от удаленных независимо движущихся объектов — источников информации. На каждом узле обработки радиолокационной информации и на каждом командном пункте управления средствами ПВО должны были применяться объединенные в локальную сеть графические терминалы различных типов для визуализации воздушной обстановки и обеспечения эффективной работы и взаимодействия оперативного и командного составов ПВО. Время отклика при этом должно было измеряться долями секунды. Система должна была базироваться на совокупности транспортабельных (способных к перемещению) РЛУ, обеспечивающих полное покрытие зоны радиолокационного обнаружения различных динамических объектов ***во всем воздушном пространстве страны.***

В сферу исследований и разработок вошел новый широкий класс компьютерных систем, в которых основными компонентами и источниками информации являлись траектории движения динамических объектов. В ходе выполнения работ нужно было создать комплексы программ для обработки информации в реальном времени. Компьютерные сети должны были обеспечивать обмен и обобщение информации о динамических объектах и их траекториях для групп РЛУ с тем, чтобы обеспечить непрерывное сопровождение объектов.

В результате следовало решить ряд ***научно-технических задач*** и создать средства для их реализации, в том числе:

- провести анализ существующих и разработать ***принципиально новые операционные системы реального времени*** для синхронизации и управления решением разнородных задач;
- разработать комплексы алгоритмов и программ для обработки радиолокационной информации в реальном времени о движущихся воздушных объектах;
- ***разработать телекоммуникационные сети для транспортировки информации между РЛУ*** о траекториях движения координат объектов;
- разработать методы и средства для тестирования в реальном времени и испытания корректности функционирования сложных программных комплексов обработки и транспортировки радиолокационной информации;
- исследовать и ***обеспечить необходимую производительность и использование ограниченных ресурсов компьютеров и каналов связи*** программными комплексами обработки и транспортировки радиолокационной информации между РЛУ в реальном времени.

К созданию такой системы были привлечены несколько сотен сотрудников разной квалификации, которые к концу 1960-х гг. разработали и испытали полный опытный образец РЛУ и телекоммуникационной сети системы ПВО. Далее изложены основные принципиальные научно-технические задачи, которые впервые в отечественной практике были решены при создании этой системы. Следует еще раз подчеркнуть, что они отрабатывались при очень ограниченных вычислительных ресурсах компьютеров и возможностей, существовавших в 1960-е гг. телекоммуникационных сетей.

Создание операционных систем для управления решением разнородных задач в режиме реального времени

В **1958—1959 гг.** в НИИ-5 начались активные работы по применению ЭВМ для решения оперативных задач, исходная информация которых поступала от нескольких несинхронных, независимых источников. Эти источники различались по степени важности и скорости решения этих задач, а также по величине допустимого интервала времени, после которого результаты вычислений становятся доступны потребителям (времени запаздывания). Процессы решения задач должны были укладываться в определенные этими ограничениями интервалы времени с учетом реальной производительности применяемых компьютеров.

Задачи, которые возникали в связи с потребностями в новых функциональных возможностях создаваемых систем, быстро усложнялись, увеличивались необходимые для их реализации производительности вычис-

лительных машин. Для решения таких задач в режиме реального времени на некоторых предприятиях оборонной промышленности началась разработка специализированных средств автоматизации управления вычислительными процессами и создание на их основе операционных систем реального времени. Необходимо было оптимизировать использование производительности ЭВМ в сложных ситуациях online — обработки несинхронных потоков информации из внешней среды о различных объектах, при разнообразии очередности и длительностей выполнения отдельных задач. Значительные трудности реализации операционных систем реального времени **в начале 1960-х гг.** состояли в ***неопределенности параметров временных динамических процессов*** поступления, длительности обработки и выдачи информации потребителям, а также в ограниченном объеме ресурсов, которые можно было выделить для размещения программ операционной системы.

Все процессы решения задач должны были укладываться в те ограничения, которые налагались производительностью компьютеров, доступных разработчикам. К сожалению, в отечественной и зарубежной литературе тех лет практически ***отсутствовали публикации по методам и средствам организации операционных систем реального времени***. Все инженерные разработки и исследования в этой области проводились как секретные, преимущественно на оборонных предприятиях при выполнении конкретных проектов.

Первые операционные системы реального времени разрабатывались на основе инженерной интуиции и эвристических подходов, без использования каких-либо аналитических оценок и формальных моделей. Для эффективной реализации требований к функциональным возможностям системы впоследствии были исследованы математические и статистические модели процессов, протекающих в конкретных операционных системах реального времени. Эти исследования базировались на достаточно развитой теории массового обслуживания и теории расписаний, которые прошли апробацию в различных инженерных областях.

Накопление и обобщение опыта и знаний о процессах и системах для решения отмеченных выше задач позволили сформулировать следующие ***требования к операционным системам ЭВМ реального времени*** для автоматизации обработки информации о динамических объектах в сложных системах:

- в процессах управления решением подобных задач необходимо использовать ***единое глобальное реальное время*** систем управления и обработки информации о динамических объектах, а также об изменениях внешней среды;
- потоки сообщений для обработки данных из внешней среды могут быть ***независимыми, несинхронными, различными по интенсивности***, содержанию, реальному времени их формирования и поступления для обработки;
- информация в сообщениях от источников и объектов внешней среды должна содержать реальное время, к которому относятся характеристики сообщений, их координаты и текущие состояния объектов;
- реальное время реализации отдельных функций системы должно эффективно упорядочиваться в соответствии с их приоритетами, установленной дисциплиной диспетчеризации, определенной при создании системы, а также с реальным временем приема информации из внешней среды;
- алгоритмы и программы на их основе, которые реализуют отдельные функции системы, могут различаться по времени их реализации и важности для пользователей, они должны включать и использовать текущее и расчетное реальное время результатов решения и исходной информации;
- для эффективного использования ограниченных ресурсов производительности и оперативной памяти компьютера целесообразно ***применять приоритеты и прерывания исполнения некоторых программ*** в соответствии с выбранными дисциплинами реализации различных функций системы;
- информация и сообщения для потребителей и внешней среды ***могут выдаваться асинхронно***, в соответствии с установленной дисциплиной, и содержать значения реального времени, которому соответствуют обработанные данные в сообщениях.

К сожалению, такие классы задач и операционных систем реального времени **до конца 1970-х гг.** не были востребованы в научных и учебных организациях. Как следствие, операционные системы этого класса развивались в основном усилиями специалистов оборонных предприятий.

Создание комплексов программ для обработки и передачи в реальном времени радиолокационной информации в распределенной компьютерной сети

Отсутствие в **1960-е гг.** зарубежной информации о разработках систем, подобных рассматриваемой в настоящей публикации, обеспечивало оригинальность отечественных технических решений. Как следствие, в ряде случаев такие решения оказались новыми и достаточно универсальными для территориальных вычислительных систем реального времени различного назначения. Обработка информации о воздушных объектах от радиолокационной станции (РЛС) кругового обзора состояла в отображении их в реальном времени на графических дисплеях операторов и формировании в ЭВМ траекторий их движения по этим данным. Оператор с помощью маркера фиксировал пространственные координаты каждого объекта на дисплее и вводил их в компьютер. По двум последовательным обнаружениям объекта определялось начало его трассы и первичные параметры движения объекта — скорость и курс. При последующих обнаружениях объекта РЛС оператор корректировал и уточнял на индикаторах его траекторию. Координаты и параметры движения объекта пересчитывались в прямоугольную систему координат, сглаживались с учетом накопленной информации

и передавались для использования на командные пункты управления активными средствами ПВО. Основные сложности обработки траекторий движения воздушных объектов возникали при их маневрировании или при движении плотных групп объектов, особенно в условиях наличия помех, затрудняющих обнаружение.

Каждая РЛС имела ограниченную зону наблюдения о воздушных объектах. Воздушный объект ***мог наблюдаться одновременно несколькими РЛС или не наблюдаться ни одной из них***. Для успешного управления активными средствами ПВО необходимо было обеспечивать возможности непрерывного использования информации о таких объектах и формирования их траекторий движения в широком воздушном пространстве. Для этого данными о трассах воздушных объектов по телекодовым каналам связи обменивались соседние РЛУ, и при последовательном их наблюдении координаты каждого объекта обобщались в единую трассу. Один и тот же воздушный объект мог фиксироваться в пространстве в различные моменты реального времени, которое передавалось в составе сообщений на другие соседние РЛУ. Кроме того, сообщения могли задерживаться при ожидании в очереди для передачи в телекоммуникационные каналы связи и поступать для обобщения с измененным реальным временем.

Достоверность координат и параметров движения одного и того же объекта от различных РЛУ могла значительно различаться, что приходилось учитывать при их обобщении в единую траекторию. Эти обобщенные трассы следовало идентифицировать и сохранять их номера во всей зоне наблюдения объекта. Однако при этом могли происходить разрывы траекторий, когда объекты не наблюдались ни одним источником в сети. Точность обобщения трасс ухудшалась при сопровождении групп объектов и при пересечении их траекторий.

Для обработки радиолокационной информации **с начала 1960-х гг.** использовалась одна из первых специализированная вычислительная машина на полупроводниках 5Э89 (Я. А. Хетагуров). Эта машина обладала скромными вычислительными ресурсами и параметрами, хотя имела неплохое по тем временам быстродействие, которое составляло около 60 тыс. операций в секунду. Память команд этой ЭВМ насчитывала 14 тыс. 18-разрядных слов. Ее не хватало на решение перманентно появляющихся новых задач, что было необходимо для качественной автоматизации обработки радиолокационной информации в реальном времени. Программисты экономили каждый бит оперативной памяти и памяти команд, что допускалось соответствующей архитектурой системы команд специализированной ЭВМ.

Разработка программ проводилась в машинном коде специализированной ЭВМ. Для отладки программ были необходимы тесты, отражающие динамику движения объектов в воздушном пространстве и действия операторов. Однако ограниченные ресурсы ЭВМ не позволяли выделить необходимую память и производительность для генерации тестов. Размеры ресурсов, необходимые для реализации вспомогательных задач, оказались соизмеримы с теми, которые требовались для решения задач, поддерживающих основные функции системы. Для обеспечения взаимодействия РЛУ и обмена информацией между ними была создана телекоммуникационная сеть на базе выделенных каналов телефонной сети связи. Структура сообщений в сети о наблюдаемых воздушных объектах и командах оперативного состава РЛУ была унифицирована, кодировалась и декодировалась в специальных устройствах, сопряженных с компьютерами РЛУ. Испытания проходили очень тяжело еще и потому, что разработчики впервые встретились с очень сложными программами реального времени. По этой причине работы по тестированию, отладке и испытаниям программ РЛУ и системы проводились непрерывно почти четыре года.

Тестирование и опытная эксплуатация программных комплексов обработки радиолокационной информации и телекоммуникационных сетей ПВО в реальном времени

Реальные полеты самолетов в целях проведения экспериментов с опытным образцом системы были очень дороги и проходили недостаточно часто. Как следствие, число тестовых вариантов такого натурного моделирования для испытаний оставалось небольшим. Это обстоятельство способствовало тому, что в программах и аппаратуре оставалось значительное число ошибок. Альтернативой тестированию в режиме физического моделирования стало создание и применение математических моделей на ЭВМ. Такие модели имитировали внешнюю среду движения воздушных объектов для генерации тестов и организации режимов функционирования тестируемых комплексов программ в реальном времени.

В 1964 г. на полигоне при натурных испытаниях алгоритмов и комплексов программ обработки радиолокационной информации впервые началась проверка функционирования сложных ***комплексов программ реального времени в телекоммуникационной сети, тестирование и отладка которых требовала соответствующей технологии.*** Это привело к принципиальному изменению методов тестирования, при которых динамика формирования тестов и результаты испытаний движущихся воздушных объектов должны были по времени соответствовать текущим состояниям вычислений в подлежащих проверке комплексах программ РЛУ. Для решения конкретных вычислительных задач тесты обычно готовились независимо от процессов разработки проверяемых программ. По этой причине реальное время их подготовки не влияло на результаты получаемых вычислений. Они могли быть введены в ЭВМ одновременно или независимо от использовавших их программ. Однако для комп-

лексов программ системы ПВО необходимо было их исполнение связать с реальным временем поступления зависящих от времени (динамических) тестов движущихся объектов, а также с текущим состоянием внешней среды. В эти годы генерация таких динамических тестов от внешних объектов на компьютерах РЛУ не представлялась возможной в силу ограниченности их вычислительных ресурсов. Как следствие, потребовалось применять независимые большие универсальные ЭВМ, ***отделив во времени и в пространстве*** процессы формирования динамических тестов движущихся воздушных объектов от моментов их использования.

В **1965 г.** для имитации тестов от движущихся объектов внешней среды в реальном времени были разработаны программы формирования магнитофильмов на универсальной ЭВМ М-20 (Ю. Г. Корольков). На этой машине предварительно формировались и записывались на специализированных магнитофонах наборы тестов. Такие тесты представляли в динамике разнообразные ситуации воздушной обстановки и движение объектов, с регистрацией в них значений реального времени поступления сообщений и координат объектов. Для траекторий движения воздушных объектов можно было задавать различные маршруты и маневры в реальном времени. Это существенно ускорило подготовку тестов и динамическую отладку комплекса программ обработки радиолокационной информации в сложных ситуациях воздушной обстановки.

Программная имитация динамических тестов внешней среды на ЭВМ системы в реальном времени позволила:

- проводить длительную непрерывную генерацию имитируемых данных для определения характеристик функционирования комплекса программ в широком диапазоне изменения условий и параметров экспериментов, что зачастую невозможно при использовании реальных объектов;
- расширять диапазоны характеристик имитируемых движущихся объектов за пределы реально существующих или доступных натурных источников данных, а также генерировать динамические потоки информации, отражающие перспективные характеристики создаваемых систем и объектов внешней среды;
- создавать тестовые данные, соответствующие критическим или опасным ситуациям функционирования и движения объектов внешней среды, которые невозможно или рискованно реализовать при натурных экспериментах;
- обеспечивать высокую повторяемость имитируемых данных при заданных условиях их генерации и возможность прекращения или приостановки имитации на любых фазах моделирования внешней среды.

В **конце 1969 г.** были ***успешно завершены государственные испытания опытного образца радиолокационного узла и фрагмента телекоммуникационной сети ПВО***, а также принято решение о широком внедрении системы в оборонный комплекс страны. Она обеспечила информацией о воздушной обстановке активные средства ПВО, что исключило возможность несанкционированного пролета воздушных объектов противника над территорией страны. Компоненты системы серийно производились и модернизировались оборонной промышленностью до конца восьмидесятых годов прошлого века. Свыше ста РЛУ, объединенных в телекоммуникационной сети, и соответствующее число командных пунктов ПВО были размещены и эксплуатировались на территории страны.

Заключение

В сферу научных исследований и промышленных разработок **в 1960-е гг.** впервые в стране вошел и был апробирован ***новый широкий класс распределенных компьютерных систем*** реального времени. Оборонная промышленность Советского Союза оказалась той базой, которая стимулировала бурное развитие отечественной вычислительной техники и программирования в 1950—1960-е гг. Практическое отсутствие в те годы информации о военных разработках зарубежных фирм определило ***независимое и оригинальное решение ряда сложнейших оборонных задач*** с использованием систем реального времени при очень ограниченных вычислительных ресурсах таких систем. Основные публикации по истории отечественной вычислительной техники были сосредоточены на особенностях и характеристиках аппаратуры ЭВМ. В этих работах немало внимания уделялось созданию сложных многофункциональных комплексов программ, автоматизации программирования отдельных их компонентов. Лишь изредка упоминались некоторые средства программной инженерии, поддерживающие процессы тестирования, комплексирования, документирования и сертификации.

Публикации по истории отечественной вычислительной техники и инженерии программ в интересах оборонных систем, для управления в области авиации, космического комплекса, атомной энергетики и для других целей в последнее время рассекречены. Многие из них представлены в Интернете, однако они описывают разные стороны отдельных проектов, отражают различные точки зрения и взгляды авторов, что вполне естественно. Вместе с тем история таких проектов ***имеет много достойных страниц***, которые могут представлять интерес для нынешних специалистов с позиции анализа, оценки и выбора эффективных подходов, которые использовались для их реализации. Один из примеров такой системы представлен в настоящей статье. Ее содержание не только имеет самостоятельную ценность в историко-публицистическом, познавательном плане. Используемые при создании сложной, национально значимой системы организационно-технические решения, методы и средства могут служить ***ориентирами*** для современных специалистов, решающих подобные задачи.

Èòîãè 8-é Ìåæäóíàðîäíîé íàó÷íî-ïðàêòè÷åñêîé êîíôåðåíöèè «Ðàçðàáîòêà ÏÎ/CEE-SECR 2012»

1—2 ноября 2012 г. в Москве прошла 8-я независимая Международная конференция «Разработка ПО» — Central & Eastern European Software Engineering Conference in Russia (CEE-SECR 2012). В программу двух дней вошли более ста докладов и панельных дискуссий. Накануне конференции, 31 октября, прошли мастер-классы.

Местом проведения вновь стал Центр новых технологий «Digital October». Более 700 участников из 15 стран мира — программисты, инженеры по качеству, архитекторы, аналитики, научные сотрудники и руководители всех уровней заполнили Центр. В холлах шли оживленные беседы, а некоторые доклады

вызывали такой интерес, что в залах не хватало места всем желающим.

Из двух залов велась онлайн-трансляция, значительно расширяя аудиторию SECR: за время конференции на сайте побывало более 5000 человек.

Конференцию этого года по праву можно назвать самой насыщенной за всю историю SECR. Время проведения сократилось на один день, но число докладов не уменьшилось: они шли одновременно в четырех залах и на площадке Open Space, что дало возможность слушателям выбирать между пятью выступлениями.

По отзывам участников, наибольший интерес вызвали:

- дискуссия «Сохранится ли профессия программиста?» — модераторы: Вячеслав Нестеров (EMC), Николай Пунтиков (First Line Software);
- доклад «Типичные ошибки стартапов» — Сергей Белоусов (Runa Capital);
- доклад «Обучение коду в мире онлайн» — Бенджамин Б. Бедерсон (University of Maryland);
- доклад «Как финансировать компанию, не продав душу» — Дмитрий Дубограев (Femida.us);
- доклад «Моделирование и облачные вычисления» — Ричард Соули (OMG) и другие.

На церемонии закрытия была вручена Премия Бертрана Мейера за лучший исследовательский доклад. Победителями стали Виталий Трифанов и Дмитрий Цителов из компании DevExperts с докладом «Динамическое обнаружение гонок в многопоточных Java-программах». Премия была учреждена Бертраном Мейером, создателем языка Эйфель, членом Программного комитета SECR и заведующим кафедрами в ETH и НИУ ИТМО.

В этом году конференция состоялась благодаря поддержке Российской Венчурной Компании, Epam Systems, Intel, T-Systems, Microsoft, Luxoft, Deutsche Bank, First Line Software, EMC, Oracle, Jet Brains, Parallels, Microfocus, Infostroy. Некоммерческое Партнерство РУССОФТ второй год подряд проводит ежегодное собрание в рамках конференции; активно поддержали конференцию и АП КИТ, а также Agile Russia, РАСПО, РАЭК, ISDEF, OMG, Cloud Standards Customer Council и многие другие.

Статьи на основе лучших докладов конференции будут опубликованы в нашем журнале в 2013 г.

Презентации, видеозаписи и фотографии доступны на сайте www.secr.ru.

О конференции «Разработка ПО/CEE-SECR»

Разработка ПО/CEE-SECR (www.secr.ru) — ежегодная независимая конференция, участниками которой становятся разработчики ПО, исследователи, инженеры-практики, лидеры общественного мнения, предприниматели. За годы работы конференция CEE-SECR завоевала заслуженный авторитет и традиционно поддерживается мировыми лидерами индустрии высоких технологий.

В состав программного комитета конференции входят эксперты мирового уровня, представляющие как ИТ-индустрию, так и научно-исследовательские организации. Среди них: профессор СпбГУ и генеральный директор Ланит-Терком Андрей Терехов, Академик РАН Виктор Иванников, генеральный директор Российской Венчурной Компании Игорь Агамирзян, соучредитель и главный архитектор CUSTIS Максим Цепков и многие другие.

1—5 апреля 2013 г. в г. Челябинске в ЮУрГУ (НИУ)
состоится **Международная научная конференция**

«Параллельные вычислительные технологии» (ПаВТ'2013)

Тематика конференции охватывает следующие направления:

♦ *Технологии параллельных и распределенных вычислений*. Универсальные высокоуровневые языки и библиотеки для написания параллельных программ, а также методы, алгоритмы и инструменты их анализа, отладки и оптимизации.

♦ *Грид и облачные вычисления*. Программное обеспечение промежуточного слоя в грид-системах. Универсальные технологии и системы построения облачных сервисов. Экономика грид и облачных вычислений.

♦ *Перспективные многопроцессорные архитектуры*. Реконфигурируемые и гибридные многопроцессорные системы. Новые многопроцессорные архитектуры. Многопроцессорные системы с многоядерными ускорителями GPU, MIC и др.

♦ *Параллельные и распределенные системы баз данных*. Разработка параллельных и распределенных систем управления базами данных (СУБД) для многопроцессорных (многоядерных) вычислительных систем и грид. Методы и алгоритмы параллельной обработки и интеллектуальный анализ баз данных.

♦ *Управление, администрирование, мониторинг и тестирование многопроцессорных систем*. Методы, алгоритмы и инструменты управления, администрирования и мониторинга многопроцессорных вычислительных систем. Разработка методик тестирования для оценки производительности многопроцессорных систем.

♦ *Вычислительная математика*. Решение вычислительно трудоемких математических задач с помощью многопроцессорных вычислительных систем. Применение многопроцессорных вычислительных систем для решения задач математической экономики и оптимизации.

♦ *Вычислительная физика*. Высокопроизводительное моделирование физических процессов.

♦ *Вычислительная химия*. Высокопроизводительное моделирование химических процессов.

♦ *Гидро-газодинамика и теплообмен*. Теория и практика решения задач вычислительной гидро- и газодинамики на суперкомпьютерах и в распределенных вычислительных средах.

♦ *Высоконелинейные и быстротекущие процессы в задачах механики*. Применение суперкомпьютерных технологий для анализа высоконелинейных и быстротекущих процессов в задачах механики твердого и жидкого тела.

♦ *Биоинформатика*. Высокопроизводительные вычисления в биоинженерии и биоинформатике.

♦ *Нанотехнологии*. Высокопроизводительные вычисления в моделирования новых нано и композитных материалов.

♦ *Климат и экология*. Применение суперкомпьютерных технологий для моделирования климатических процессов.

♦ *Криптография*. Параллельные методы и алгоритмы шифрования и обеспечения конфиденциальности и аутентичности информации.

♦ *Визуализация*. Параллельные методы и алгоритмы визуализации результатов научных вычислений. Визуализация в грид-средах.

♦ *Компьютерная алгебра*. Параллельные методы и алгоритмы символьных вычислений.

♦ *Суперкомпьютерные НОЦ*. Опыт формирования научно-образовательных центров, осуществляющих популяризацию и внедрение суперкомпьютерных технологий в образовании, науке и технике. Методика преподавания параллельных вычислительных технологий.

Официальный сайт конференции: http://agora.guru.ru/pavt2013/

Óêàçàòåëü ñòàòåé, îïóáëèêîâàííûõ â æóðíàëå "Ïðîãðàììíàÿ èíæåíåðèÿ" â 2012 ã.

CONTENTS

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4

Дизайнер *Т. Н. Погорелова*. Технический редактор *Е. М. Патрушева*. Корректор *Е. В. Комиссарова*

Сдано в набор 10.10.2012 г. Подписано в печать 22.11.2012 г. Формат 60×88 1/8. Заказ PI912
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
105120, г. Москва, ул. Нижняя Сыромятническая, д. 5/7, стр. 2, офис 2.