АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ

№ 4 1986

АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ ДЛЯ НАУЧНЫХ ИССЛЕДОВАНИЙ

УДК 681.3.06

А. Н. АЛЕШАЕВ, С. Д. БЕЛОВ, Б. В. ЛЕВИЧЕВ,
Г. С. ПИСКУНОВ, С. В. ТАРАРЫШКИН
(Новосибирск)

ПОСТРОЕНИЕ РАСПРЕДЕЛЕННЫХ СИСТЕМ УПРАВЛЕНИЯ КРУПНЫМИ ЭЛЕКТРОФИЗИЧЕСКИМИ УСТАНОВКАМИ НА БАЗЕ СЕТЕЙ СПЕЦИАЛИЗИРОВАННЫХ МИКРОЭВМ В ИЯФ СО АН СССР И ИХ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Интенсивное развитие микроэлектроники — мощный стимул, в значительной степени определяющий прогресс в самых различных областях человеческой деятельности, в том числе в организации физического эксперимента. В частности, развитие средств вычислительной техники на основе СБИС обусловливает переход от традиционных систем информационного обеспечения и управления экспериментом на базе одной-двух универсальных ЭВМ к построению и эксплуатации многомашинных распределенных систем на базе специализированных микропроцессорных устройств.

Можно проследить эволюцию систем управления и их программного обеспечения по мере развития этих систем в ускорительных лабораториях ИЯФ СО АН СССР. В системе, впервые обеспечившей управление накопительным кольцом ВЭПП-3, использовалась ЭВМ «Минск-22» с управляющей программой, написанной в кодах машины. Последующие системы, работающие уже на ЭВМ «Одра-1304», были рассчитаны на управление сразу несколькими установками. Их программное обеспечение представляло собой «монолитную» сборку, т. е. одну большую программу, работающую в диалоговом режиме. Активное использование диалога позволило оперативно менять большое количество параметров, определяющих жестко заданный алгоритм управления. В дальнейшем «входной язык» этой системы развился до такой степени, что появилась возможность говорить действительно о языке и использовать его для написания простых управляющих программ — процессов.

Интерпретаторы такого рода очень гибки, позволяют ввести режим мультипрограммирования, обеспечить работу многомашинного комплекса и поэтому широко используются во многих системах управления (например, NODAL [1]). Гибкость интерпретаторов дает возможность ввести в язык дополнительные средства, сильно расширяющие возможности системы: организацию взаимодействия процессов, работу в разнесенных машинах. К сожалению, при этом оказывается, что доступ к таким средствам становится ограниченным рамками выбранного языка, который в силу своей «узкой» специализации может оказаться неадекватным возникающим новым задачам. Кроме того, интерпретация весьма существенно замедляет выполнение рабочих программ.

Таким образом, реализация в рамках единого интерпретируемого языка операционной среды большой распределенной системы управления неизбежно ограничивает возможности развития системы. Харак-

терно, что в большинстве приложений ЭВМ явно определилась тенденция к отказу от чрезмерной универсализации программных средств, будь то операционные системы или языки программирования. Поэтому нами был выбран иной путь — построение мультипрограммной операционной системы (ОС), в среде которой выполняются собственно управляющие программы [2]. При этом исключается привязка возможностей системы к одному конкретному языку. В этих условиях могут использоваться как интерпретирующие, так и компилирующие системы программирования.

Общепризнано, что эффективность системы управления определяется не только и даже не столько такими «престижными» характеристиками ее отдельных компонент, как быстродействие центральных процессоров применяемых ЭВМ, наличие «фирменных» операционных систем, пропускная способность машинных каналов и шин. Возможности системы в целом определяются в большей степени не параметрами образующих ее элементов и подсистем, а свойствами ее структуры, развитостью и регулярностью системообразующих связей между компонентами каждого уровня, а также рациональностью и эффективностью интерфейсов между всеми уровнями системы: как на уровне сопряжения процессора с периферийной аппаратурой, так и на более высоких уровнях операционных систем (интерфейсы межпроцессорных и межпрограммных взаимодействий, интерфейс с человеком-оператором). Разумеется, высокие эксплуатационные характеристики системы могут быть в полной мере реализованы лишь при использовании качественных аппаратных средств, включающих разветвленную систему передачи данных и представительный набор периферийных устройств для управления и сбора информации.

Возрастающие требования к развитым системам управления и новые возможности, предоставляемые разработчикам аппаратуры современной элементной базой, приводят к существенному увеличению функциональной насыщенности периферийных устройств, а унификация средств взаимодействия с устройствами диктует необходимость использования модульной организации систем. В качестве стандарта, применяемого в настоящее время в институте для построения систем управления, используется стандарт КАМАК.

Стандарт КАМАК предназначался в основном для систем регистрации в физике высоких энергий, и его применение в системах управления обычно сопряжено с некоторыми трудностями. Так, высокая степень интерактивности взаимодействия процессора и периферийного оборудования в КАМАК-системах, т. е. ориентация таких систем на программный обмен, существенно усложняет организацию передачи данных по каналам прямого доступа к памяти. Программный обмен предоставляет весьма гибкие средства для организации передачи; в то же время в современных мини-ЭВМ, работающих в многопрограммном режиме, организация программного обмена данными с аппаратурой через распространенные крейт-контроллеры приводит к значительному увеличению загрузки процессора и не позволяет в полной мере использовать высокую пропускную способность КАМАК-магистрали. Для сокращения количества необходимых контекстных переключений при передаче длинного блока данных и разгрузки процессора предпочтительнее использование обмена по прямому доступу.

Каналы прямого доступа ориентированы на передачу длинных блоков данных, возникающих при работе с современными функционально и информационно насыщенными КАМАК-модулями; однако при этом протоколы взаимодействия аппаратуры канала и периферийных устройств весьма примитивны. В то же время характер взаимодействия с такими устройствами при передаче каждого слова данных достаточно прост, что определяет возможность работы с этими устройствами по прямому доступу, но требует разработки специальных контроллеров.

Для решения этих проблем предлагалось выделять отдельную спе-

циализированную микроЭВМ, что приводит к разделению всей системы на два уровня. В подобных системах взаимодействие с аппаратурой, предварительная обработка, форматирование и частично визуализация информации осуществляются на нижнем уровне, а обмен с главной ЭВМ ведется большими массивами уже готовых данных. В качестве такой микроЭВМ часто используется отдельная машина, либо микропроцессор встраивается в контроллер. Подобные интеллектуальные контроллеры применяются во многих лабораториях как в СССР, так и за рубежом.

Несмотря на наличие определенных преимуществ, применение интеллектуальных контроллеров зачастую осложняется следующими обстоятельствами:

микроЭВМ на базе микропроцессора в составе крейт-контроллера обычно обладает весьма ограниченными аппаратными ресурсами, что требует программирования ее на языке ассемблерного уровня;

сложность и функциональная насыщенность аппаратуры требует организации мультипрограммного режима использования микроЭВМ; этот режим обычно обеспечивается на уровне ОС, но в данном случае его приходится организовывать «вручную»;

необходимость применения незнакомых языков низкого уровня и сложность работы в «голой» машине, недостаточность средств отладки программ не позволяют привлечь инженеров и физиков для разработки программного обеспечения таких контроллеров;

единая, «монолитная» программа-сборка на языке ассемблерного уровня, реализующая множество функций, трудна для понимания, сопровождения и модификации;

из-за применения сильно разнящихся ЭВМ на верхнем и нижнем уровнях системы управления становится затруднительным информационное совмещение этих уровней.

Чтобы обойти перечисленные трудности, в ИЯФ СО АН СССР в 1981 г. было предложено создавать интеллектуальные крейт-контроллеры, реализующие системы команд центральных ЭВМ, т. е. ЭВМ верхнего уровня. Ввиду полной программной совместимости ЭВМ верхнего и нижнего уровней отпадает необходимость в кросс-системах для подготовки программ, появляется возможность использования единого языка высокого уровня. Подобные микроЭВМ, имеющие архитектуру системы «Одра-1300», были разработаны и успешно применены в институте. Данная микроЭВМ, получившая условное название «Одренок» [3], представляет собой автономный контроллер крейта, выполненный в КАМАК-модуле шириной 2 М. Связь с ЭВМ верхнего уровня осуществляется при помощи отдельного интерфейсного КАМАК-модуля, имеющего простой протокол обмена, использующего стандартные КАМАК-функции и позволяющего организовать работу «Одренка» с обычными ЭВМ, образующими верхний уровень системы.

Высокая производительность процессора и значительный объем оперативной памяти позволяют применять микроЭВМ для решения существенно более широкого круга задач, чем предполагалось при его проектировании. Так, кроме использования «Одренка» в системах управления оказалось возможным применять его в качестве мощного персонального вычислителя — рабочей станции с развитыми графическими возможностями — для организации рабочего места разработчика и наладчика цифровой аппаратуры, для автоматизации небольших экспериментов.

Созданные микроЭВМ легко и естественно включились в ранее существовавшие в институте централизованные системы управления. Однако в таких системах представляется затруднительным разделить функции комплекса управляющих программ на достаточно содержательные фрагменты, пригодные для выделения в отдельные процессоры, из-за существования сильных информационных зависимостей между всеми подсистемами. Вследствие этого данные микроЭВМ использовались в

качестве вспомогательных процессоров нижнего уровня, занятых решением достаточно изолированных и информационно замкнутых задач. При этом не потребовалась модификация использовавшихся операционных систем; взаимодействие между основным и вспомогательными процессорами осуществлялось на уровне прикладных программ. Так, на комплексе ВЭПП-4 были реализованы следующие подсистемы:

измерения и отображения результатов измерений светимости, частоты обращения пучка и бетатронных частот накопителя, параметров орбиты;

измерения с помощью микроканальных датчиков положения пучка в электронно-оптическом канале позитронного инжектора в семи сечениях канала с частотой работы инжектора;

контроля радиационной обстановки на комплексе ВЭПП-4 и экспериментальных станциях синхротронного излучения (50 точек контроля), при этом включение «Одренка» в эту подсистему обеспечило непрерывность контроля.

По мере накопления опыта использования «Одренка» в прежних системах управления было отмечено, что при таком подходе осложняется дальнейшее их развитие, и поэтому необходим переход к распределенным системам. Создание же распределенной системы выдвигает новые требования к системному программному обеспечению и приводит к некоторому перераспределению «ответственности» за поддержку тех или иных системных функций. Так, например, связи между программами в разных процессорах должны обеспечиваться единообразно на уровне операционных систем, которые также организуют совместный доступ к общим базам данных и обеспечивают синхронизацию процессов в масштабе всей сети.

Иерархичность в распределенных системах может сохраниться, однако функции центральной ЭВМ определенным образом редуцируются до функций файловой машины сети, обеспечивающей периферийные ЭВМ такими ресурсами, как дисковая память, терминалы, принтеры, а также возможностью обращения к ленточному архиву. В этом случае выбор топологии сети, объединяющей вычислительные средства системы, оказывает меньшее влияние на архитектуру системы в целом. Так, нами выбрана звездообразная конфигурация как более простая в реализации, хотя возможны и варианты с кольцевой или с шинной организацией.

Новой, не реализованной в прежних операционных системах должна быть функция обеспечения межпрограммного взаимодействия для программ, работающих в разных процессорах нижнего уровня. Желательно обеспечить при этом максимальную унификацию программных интерфейсов периферийной операционной системы для различных типов обмена: межпрограммного в пределах одного процессора, межпрограммного для разных процессоров, программных обменов как с файлами, расположенными на накопителях данного крейта (локальными файлами), так и с файлами, размещенными на носителях файловой машины.

Таким образом, программная среда, в которой выполняется программа в периферийном процессоре, отличается от среды, обеспечиваемой как большинством стандартных, фирменных ОС, так и семейством разработанных ранее в институте ОС систем управления.

В новых операционных системах существенную роль играет понятие логического канала обмена, приписанного программе. Каждая программа в периферийном процессоре может использовать до восьми таких каналов для организации взаимодействия с другими программами и для файловых обменов, ссылаясь на номер канала в каждом элементарном акте обмена. Пункты отправления и назначения передаваемой информации определяются в выполненной ранее процедуре открытия канала обмена. Такими пунктами (абонентами) могут являться как программы, так и файлы.

Можно определить два основных способа обмена данными: асинхронный, когда данные передаются по инициативе одного из абонентов вне зависимости от состояния другого (например, все файловые обмены относятся к этой категории), и синхронные, когда для реализации передачи данных необходимо, чтобы один из абонентов выставил ОС запрос «Читай», а другой — соответственно «Пиши». Механизм синхронного обмена, или симметричного рандеву, обеспечивает решение практически всех проблем мультипрограммирования, традиционно решаемых с помощью низкоуровневых примитивов типа семафоров. Кроме того, этот механизм достаточно универсален и пригоден для использования в мультипроцессорных, мультипроцессорных с общей памятью и однопроцессорных архитектурах.

При асинхронных обменах каждая программа, выступающая в роли пассивного абонента, может объявить некоторую связную область программой в сети. Аналогичные механизмы широко применялись в ОС первого поколения как обмены типа «дай — бери». Как и обычные файлы центральной файловой системы, программные файлы являются коммунальными, т. е. доступ к такому файлу разрешен произвольному количеству программ.

При синхронном обмене программа «*A*» может заявить системе о создании канала с заданным программным номером, предназначенного для монопольного использования программой «*B*». В свою очередь, программа «*B*» должна открыть канал, предназначенный для программы «*A*». После замыкания такого канала, т. е. открытия его обеими программами, может производиться обмен данными. Каналы предполагаются дуплексными, т. е. одновременно может идти передача данных в обе стороны.

Средства синхронизации, предоставляемые ОС, образуют флаговую систему. Каждая программа может объявить три последовательных слова своего адресного пространства флаговым полем, в котором ОС будет устанавливать пометки о системных событиях для данной программы. Первые два слова поля используются для пометок о LAM-запросах в крейте, о сигналах синхронизации с системного модуля — регистра прерываний — и о программно генерируемых системных событиях. Каждая программа может генерировать такое событие как имеющее локальный характер, т. е. только для своего процессора, так и в качестве глобального, общесистемного сигнала. Третье флаговое слово используется для пометок об окончании передачи по каналам, открытым программой, и о таких системных событиях, как истечение некоторого временного интервала, принятие системой к исполнению директивы, относящейся к данной программе, установка флагов, связанных с терминальными обменами, и т. д.

Программа, ожидающая некоторого системного события либо совокупности событий, выполняет экстракод, задающий три слова масок, соответствующих этим событиям. При возникновении одного или нескольких событий, интересующих программу, система выведет ее из состояния ожидания, а информация во флаговом поле позволит программе определить причину ее активации.

Как уже отмечалось, файловая система может состоять как из локальных, так и из центральных файлов. По отношению к файлу программа может выполнять обычные операции — открыть файл, приписав его к какому-то программному каналу, выполнить обмен данными, закрыть файл. При этом возможная блочная организация данных на накопителях центральной ЭВМ оказывается скрытой в ОС и адресация в

файле ведется на уровне слов, что накладывает определенные ограничения на максимальную длину файла. При 24-разрядном адресе максимальный размер файла будет порядка 48 Мбайт. Физическая структура центральных файлов локализована в управляющей программе центра, что позволяет избежать введения алгоритмов распределения дисковой памяти в периферийные ОС, существенно упрощая их.

Ввиду того что система управляет многими установками, входящими в состав комплекса, одновременно во всей системе будет работать значительное число программ, ведущих активный диалог с операторами, особенно на этапе наладки и запуска. Обеспечение каждого процессора сети несколькими терминалами для общения с программами было бы нерациональным, так как активно использоваться будут не все терминалами. Доступ к каждому процессору осуществляется через систему виртуальных терминалов: каждый незанятый терминал центра может быть приписан любой периферийной ЭВМ, что позволяет гибко и эффективно распределять терминалы между процессорами сети в соответствии с текущими требованиями. Такое назначение терминала (либо его освобождение) происходит по инициативе оператора.

Вывод на терминал практически во всех системах осуществляется одинаково; в случае терминального ввода возможны варианты. Распространены два сильно отличающихся подхода: в первом программа выставляет запрос на ввод достаточно длинного сообщения, завершающегося одним из немногих допускаемых системой (если не единственным) символом (терминатором); во втором случае программа ведет посимвольный прием информации с терминала, самостоятельно формируя при этом логически завершенное поле (например, число из нескольких цифр). При этом программа может не использовать жестко заданный набор терминаторов, а динамически переопределять его в процессе выполнения. Это делает возможным организацию очень содержательного и насыщенного диалога. Как правило, такие возможности реализуются в рамках специальных пакетов подпрограмм. Все управляющие программы написаны с использованием подобных пакетов.

Однако такой подход может привести к некоторому увеличению потока служебных обменов в сети при использовании виртуальных терминалов. Поэтому представляется необходимым ввести в систему еще один тип обмена — терминальный обмен. Каждая программа приписана к одному из терминалов, доступных выполняющему ее процессору: локальному либо терминалу центра; тем самым определен ее канал терминального обмена. Допускаются обычные операции вывода по этому каналу. Дополнительно программа может запросить:

переопределить заданный для ее терминального канала набор символов-терминаторов;

ввести с терминала последовательность символов, завершающуюся одним из определенных в данный момент символов-терминаторов;

разрешить или запретить использование при вводе функций редактирования, реализующихся на данном терминале.

Для организации работы комплекса программ как в одном процессоре централизованной системы, так и в случае распределенной системы представляется полезной возможность инициирования выполнения некоторого системного действия не только с терминала, но и по запросу произвольной программы сети. Например, это позволяет программам автоматически загружать в память, запускать друг друга, изменять при необходимости некоторые системные параметры.

В настоящее время в институте работают четыре комплекса, объединяющие по 6—15 микроЭВМ «Одренок». Часть перечисленных выше функций в этих комплексах реализована пока не на уровне ОС, а на уровне прикладных программ, и в целом системы носят экспериментальный, модельный характер. Несмотря на это они уже позволяют решать многие возникающие в институте задачи, хотя работа над полными версиями операционных систем еще не завершена.

ЛИТЕРАТУРА

1. **The NODAL** System for the SPS /M. C. Crowley-Milling, S. Shiering.— Geneva, 1978. (Preprint/CERN; 78—07).
2. **Алешаев А. Н., Белов С. Д., Левичев Б. В. и др.** Операционная система ЭВМ «Одра» для управления электрофизическими установками в ИЯФ СО АН СССР.— Новосибирск, 1980. (Препринт/АН СССР, Сиб. отд-ние, ИЯФ; 80—194).
3. **Пискунов Г. С., Тарарышкин С. В.** Двадцатичетырехразрядная ЭВМ в стандарте КАМАК.— Автометрия, 1986, № 4.

Поступила в редакцию 3 февраля 1986 г.

УДК 681.3.022

А. Г. АТАМАНЧУК, В. И. БЕНЕВЕЛЬСКИЙ, И. И. ГРАЧЕВА, Н. М. ГУЛИНА, С. Г. ДОЛГОБРОДОВ, А. Н. ЛОДКИН, П. В. НЕУСТРОЕВ, А. А. ОРЕШКИН, Е. М. ОРИЩИН, Т. С. СЕРЕБРОВА, Н. А. СЕРЕГИН, Б. Ю. СОКОЛОВСКИЙ, Е. В. ФОТЬЕВА, А. Е. ШЕВЕЛЬ

(Гатчина Ленинградской)

РЕАЛИЗАЦИЯ ЛОКАЛЬНОЙ ТЕРМИНАЛЬНОЙ СЕТИ

Введение. В экспериментах по физике высоких энергий с каждым годом растут затраты на обработку результатов измерений, что обусловлено объемом данных и сложностью алгоритмов обработки. Поскольку в процесс обработки и интерпретации результатов обработки измерений вовлечено значительное количество экспериментаторов, то возникает проблема организации эффективного доступа к вычислительным мощностям на основе широкого использования терминалов. При этом основная масса терминалов должна располагаться в тех местах, где они необходимы экспериментаторам.

В Ленинградском институте ядерной физики им. Б. П. Константинова АН СССР для обработки результатов измерений используются несколько ЭВМ, рассредоточенных по площадке института. (В данной работе речь идет об ЭВМ ЕС-1030 и ЕС-1060, расположенных на расстоянии примерно 400 м друг от друга.) Очевидные преимущества заключаются в возможности подключения любого терминала к любой из этих машин по команде с клавиатуры дисплея.

Цель данной статьи — рассмотрение конкретной реализации терминальной сети в ЛИЯФ АН СССР.

Конфигурация оборудования. Для решения поставленной задачи использовалась аппаратная конфигурация, показанная на рис. 1. Основным требованием к ней было максимально возможное использование стандартного оборудования. Специальное внимание уделялось уменьшению объема оборудования, которое ранее не применялось в ЛИЯФ. Очевидная цель состояла в том, чтобы минимизировать проблемы, связанные с эксплуатацией.

Функционирование сети терминалов. В конфигурации, представленной на рис. 1, ЭВМ СМ-4 используется в качестве программируемого связного контроллера, который осуществляет пересылку строк текста (за-