УДК 681.518:681.3.016

ОБ ОДНОЙ ТЕХНОЛОГИИ МОНИТОРИНГА И НАТУРНЫХ ИССЛЕДОВАНИЙ ЛОКАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ

А.В. Воруев, О.М. Демиденко, В.А. Никишаев, М.В. Потрашкова

Гомельский государственный университет им.Ф.Скорины, кафедра АСОИ,
г. Гомель, ул. Советская, 104. Тел. (0232) 57-88-63.
E-mail: ANG@gsu.unibel.by

Аннотация. Данная статья посвящена разработке монитора вычислительного процесса для натурных экспериментов на узле сети для дальнейшей оценки его влияния на загруженность сети и поиска рекомендаций по решению данной ситуации. Информация мониторинга будет входной для имитационной модели.

Ключевые слова. Мониторинг, процесс, моделирование, операционная система (ОС).

Целью данной работы является получение и обработка данных мониторинга вычислительной системы и дальнейший анализ полученных результатов. Требуется произвести эксперимент над вычислительной системой, затем по данным журнала системного монитора предоставить величины на вход имитационной модели, а также произвести анализ вычислительного процесса и сделать выводы об операционной обстановке в системе.

Обычно пользователи получают уведомления об аварийной ситуации через свои приложения после того, как сбой уже случился. Чтобы предсказать, в каком месте в следующий момент может возникнуть очередная проблема, и был создан монитор приложений. Несмотря на то, что эта концепция не нова — мониторы времени отклика в больших ЭВМ уже давно зарекомендовали себя с лучшей стороны, — в последние год-полтора число продуктов мониторинга приложений стремительно возросло. Отчасти это произошло благодаря всеобщему признанию Windows NT и тому, что в корпоративных сетях предпочтение все больше отдается протоколу IP.

Некоторые из этих продуктов работают в режиме фоновой службы, осуществляя замер показателей производительности локального ПК, сети и сервера в реальном времени или согласно заданному расписанию.

Контроль может вестись из трех основных точек: с клиентских машин, сервера приложений и из сети (в виде зондов RMON). Со стороны сети контроль осуществляется по протоколу SNMP. Последней новинкой в области мониторинга производительности стали облегченные агенты, устанавливаемые на клиентских машинах или серверах. Поскольку существует два типа агентов — активные и пассивные, — можно выделить и два типа мониторинга: активный, основанный на искусственной генерации передаваемых через сеть транзакций, пассивный — на регистрации событий, происходящих на локальном клиенте.

Есть различные мнения о том, где и как максимально эффективно можно осуществлять текущий контроль, но сегодняшние методы пока еще далеки от совершенства. В данной статье более детально рассматриваются пассивный и активный методы и обсуждаются преимущества и недостатки каждого из них.

Первый и, пожалуй, наиболее проработанный подход к мониторингу приложений — серверный. Статистика в этом случае собирается на серверах и клиентах и затем сопоставляется с производительностью сервера.

К сожалению, данный метод не учитывает условия эксплуатации сети, зато он имеет преимущество — централизованное измерение на сервере приложений.

Серверный метод позволяет собрать информацию, наиболее детально описывающую работу тех или иных приложений для их диагностики, настройки и прогнозирования их поведения. Приложения фирм Envive, Oracle и PeopleSoft благодаря средствам настройки позволяют учитывать особенности транзакций различного вида, например, используемых в СУБД или в серверных прикладных процессах.

Менее сложными, но в то же время и менее специализированными в области мониторинга приложений являются такие продукты, как Patrol фирмы BMC Software, Performance Monitor фирмы Computer Associates и

PerfView фирмы Hewlett-Packard. Наряду с другими измерениями они в первую очередь ориентированы на регистрацию показателей операционной системы об использовании памяти, результативности обращений к кэшу, дисковым и сетевым устройствам ввода-вывода. Их роль в мониторинге приложений приобретает все большее значение. Если исходить из правильного предположения: раз операционная среда функционирует хорошо, так же хорошо работают и приложения, то к достоинствам данных продуктов можно отнести легкость их освоения и более простую, чем у специализированных серверных продуктов, методику контроля.

И серверные методы, и подходы, основанные на операционных системах, позволяют прогнозировать работу приложений в будущем. Однако, серверный метод имеет дополнительные источники погрешностей, обусловленные сложностью сетевой инфраструктуры и большим числом промежуточных серверов.

Мониторинг транзакций первоначально выполняется зондами, которые отслеживают сетевой трафик и прогнозируют производительность приложений путем соответствующих вычислений. Помимо того, что этот пассивный метод никак не влияет на работу сервера или клиента, он позволяет производить анализ корреляции производительности сети и приложения.

Для того, чтобы текущий контроль охватывал всю сеть в целом, необходимо устанавливать зонд в каждый сегмент, что само по себе довольно проблематично для коммутируемых сетей с большим числом сегментов. Но, отчасти можно обойти подобное требование, используя стратегию размещения зондов на магистральной сети, в сегментах масштабируемого серверного комплекса и других критически важных объединяющих сегментах. Надо сказать, что такой подход не позволяет контролировать работу клиентов, зато обеспечивает централизованный мониторинг для всех важных транзакций.

Сбор и расшифровка колоссального числа пакетов в тысячах сетевых соединений - очень сложный процесс. Однако, зонды удаленного мониторинга вполне справляются с этой задачей, подтверждая действенность данного метода. Другие подходы, в которых за основу взята единовременная обработка огромного количества собранной информации, не оставляют надежды на получение своевременного отчета и тем самым дискредитируют идею оперативного мониторинга производительности.

Сторонники пассивного метода отмечают, что он обеспечивает реальные замеры производительности с учетом особенностей клиентов в данной сети — типов используемых транзакций, времени дня, времени реакции пользователя и т. д. Единственный его недостаток — это разброс длительности транзакций. Число пакетов в реальных транзакциях постоянно варьируется, поэтому не возможно будет получить точных измерений сетевой производительности. Но это не означает, что вовсе не возможно получить время ожидания. Например, пакет VitalSuite фирмы Vital Signs для вычисления времени выполнения транзакции использует пакеты с выставленным битом SYN.

Альтернативными являются продукты, использующие искусственные транзакции, которые либо запрограммированы, либо оперативно меняются пользователем и запускаются с удаленного агента по заранее намеченному графику. Это могут быть запросы ping или DNS, обращения по протоколу SMTP или такие специфические, как SQL-операторы, или определяемые пользователем транзакции. Время их выполнения известно заранее, поэтому любое отклонение от его базовой величины будет означать ухудшение производительности.

В отличие от пассивного метода, приложения, основанные на искусственных транзакциях, могут быть проверены и при отсутствии клиентов в сети. Это существенное преимущество, особенно в тех случаях, когда в распоряжении сетевого администратора в нужный момент просто нет достаточного количества реальных пользователей.

Другая модификация метода искусственной транзакции включает использование и транзакции клиента, и сервера приложения. Продукт Pegasus фирмы Ganymede Software реализует именно такой подход. Несмотря на то что генерируемые им транзакции в реальных условиях не используются, они позволяют исключить влияние сетевой инфраструктуры на замеры производительности приложений.

К сожалению, искусственные транзакции вносят дополнительную нагрузку на сеть. Это не вызывает особых проблем в локальных сетях, однако повлечет серьезные задержки в звеньях распределенных глобальных сетей.

Если хотя бы одно устройство в каждом удаленном офисе посылает искусственные транзакции на центральный узел через канал распределенной сети, такое тестирование просто парализует сеть.

Выбор наиболее подходящего способа мониторинга зависит от того, какой набор приложений вы собираетесь контролировать и какие параметры информационной системы вы хотели бы в результате усовершенствовать. Если для вас наиболее важна поддержка клиента, обрабатывающего справочные запросы, а сами транзакции являются достаточно сложными, то оптимален пассивный метод. В случае же, когда во главу угла ставится сетевая производительность, наиболее предпочтительной будет искусственная транзакция между двумя агентами.

Взяв за основу способ реализации, основанный на оценке удаленных пассивных клиентов, основной упор на текущем этапе работы был сделан на написание клиентской части мониторинга вычислительного процесса узла локальной сети. Поскольку для правильного вывода об его влиянии на производительность сети нам необходимо знать не только объем обрабатываемых им транзактов, но и его возможности по их обработке – данная клиентская часть должна обладать большим числом диагностических замеров.

Разрабатываемая система мониторинга должна быть ориентирована на какую-либо конкретную операционную систему. Так как в настоящее время наиболее распространенными являются операционные системы фирмы

Microsoft, то именно они будут использоваться в качестве платформы для системы мониторинга. В настоящее время используются следующие версии Windows – это Windows 95, Windows 98, Windows NT 4.0 и Windows 2000. Но, по своей архитектуре они делятся только на два класса – Windows 9x и NT. Для программ, работающих на уровне Win32 API (прикладной программный интерфейс), различия между этими архитектурами практически незаметны. Множество функций Win32 API из Windows 9x являются подмножеством функций настоящего Win32 API от NT.

Задачи, поставленные при разработке системы мониторинга, требуют очень тесной интеграции с операционной системой. Поэтому модуль, отвечающий за перехват событий, должен работать на уровне ядра ОС. Это возможно, если данный модуль будет написан в виде драйвера устройства. В случае с ОС Windows 95/98 это будет файл с расширением VXD, а для NT – SYS. Написание драйверов устройств является нетривиальной задачей, причем создание драйверов для NT на порядок сложнее, чем для Windows 95. В данной работе рассматриваются только операционные системы Windows 95/98, так как ядро NT представляет гораздо меньше средств по перехвату системных событий и большинство функций ядра недокументированно. Но, благодаря тому, что программа сбора статистики получает данные от драйвера, используя стандартный протокол обмена, после создания драйвера для NT не придется переписывать остальные модули системы мониторинга.

Структура программы системы мониторинга определяется решаемыми ей задачами. К этим задачам относятся:

- отслеживание системных событий (переключение потоков, обращения к диску, использование виртуальной памяти, графические операции);
- сопоставление этих операций процессам;
- периодическая запись статистических данных на диск.

Все эти задачи могли бы быть решены в рамках одного модуля – драйвера устройства. Но из-за того, что одним из принципов построения операционных систем является принцип изолированности вертикальных уровней, невозможно определить, какой именно процесс выполняет ту или иную операцию. То есть, драйверу неизвестно имя процесса и идентификатор процесса уровня 0 не равен идентификатору процесса уровня 3. Поэтому система мониторинга состоит из трех частей:

- модуль перехвата системных событий;
- модуль идентификации процессов;
- модуль сбора статистики.

Первая часть представляет собой драйвер, написанный на языке C++ с использованием библиотеки VToolsD. Данный драйвер перехватывает следующие системные сервисы: переключение потоков, файловые операции, использование виртуальной памяти, графические операции. Для хранения информации драйвер выделяет в невыгружаемой памяти буфер емкостью 2048 событий. Скорость заполнения буфера изменятся от 5 до 1500 событий в секунду. Поэтому программа сбора статистики должна своевременно считывать данные из буфера.

Модуль идентификации процессов представляет собой динамически загружаемую библиотеку, содержащую системную ловушку. Использование ловушки позволяет загрузить библиотеку в адресное пространство каждого процесса. Во время инициализации библиотеки вызывается функция получения системного идентификатора процесса, а затем этот идентификатор отсылается программе сбора статистики.

Модуль сбора статистики представляет собой программу, написанную на языке Borland Delphi 3.0. Данная программа выполняет следующие действия:

- загружает драйвер перехвата системных событий;
- регистрирует в системе ловушку, содержащую модуль идентификации процессов;
- создает файл журнала событий с уникальным именем;
- запускает отдельный программный поток, отвечающий за считывание данных из драйвера и записи их в журнал;
- отображение промежуточных данных на дисплее.

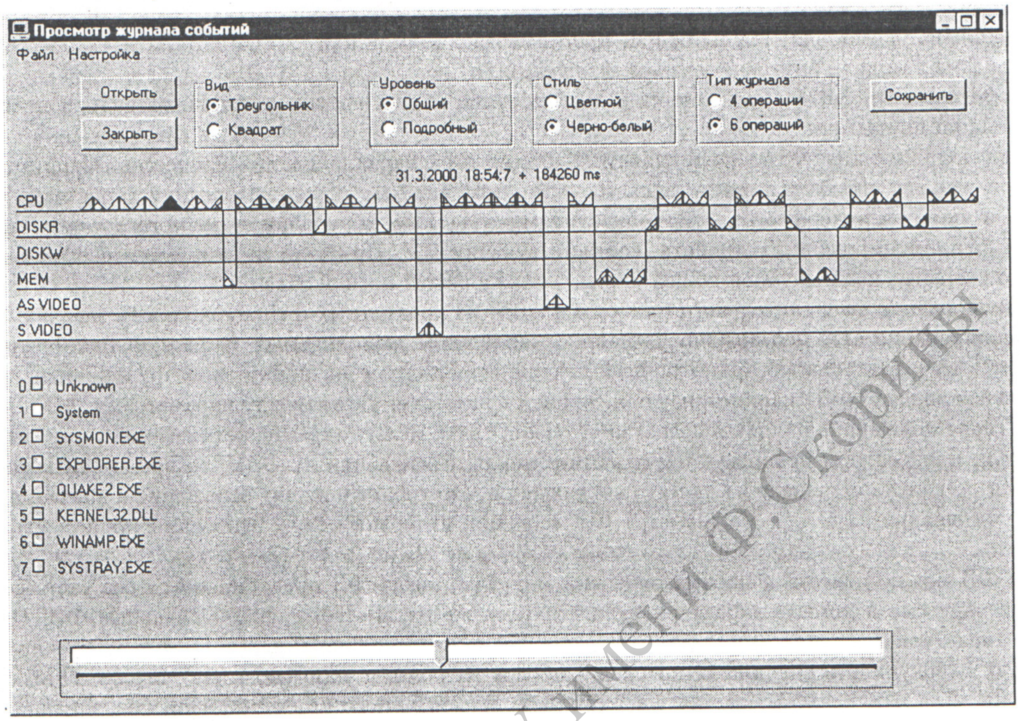
Система мониторинга реализована в виде трех программных модулей:

- Драйвер уровня ядра операционной системы. Он отвечает за перехват всех исследуемых операций, а также первоначального накопления информации в локальном буфере драйвера.
- Динамическая библиотека системной ловушки. Она предназначена для идентификации имен процессов в системе.
- Модуль сбора и записи статистики, получаемой от драйвера, на диск.

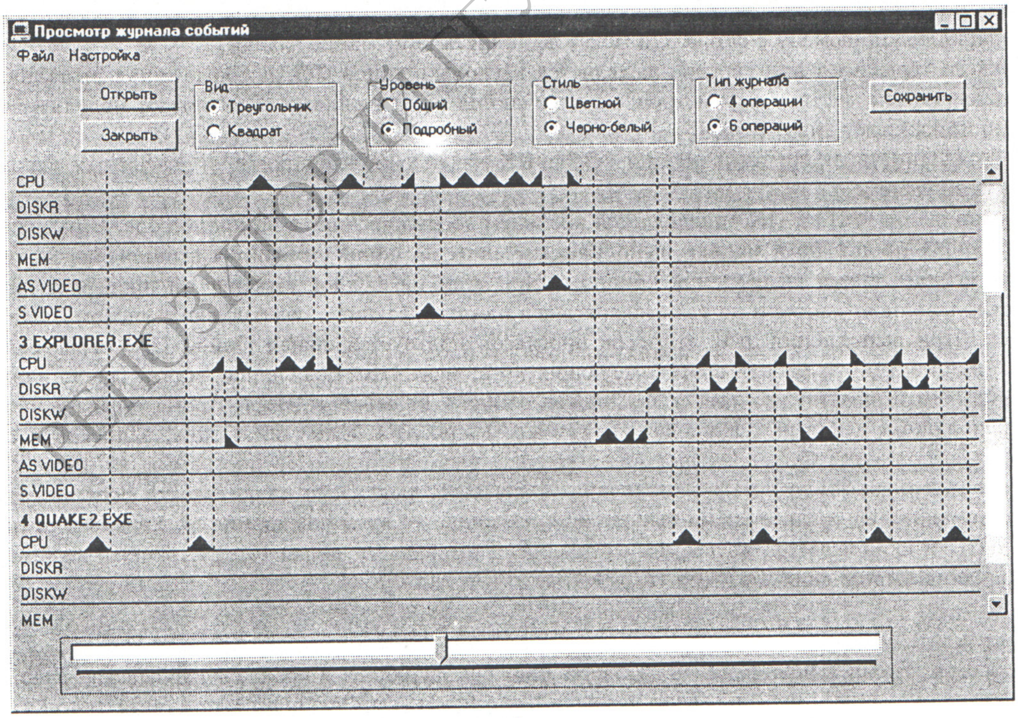
Данный монитор позволяет снять картину поведения вычислительного процесса в условиях реальных нагрузок. По результатам анализа создается компактная и достаточно полная математическая модель вычислительного процесса, на базе которой можно делать выводы об адекватности аппаратной базы той рабочей нагрузке, которая на ней выполняется.

Исследование вычислительного процесса таких ЭВМ интересно как с точки зрения разработчиков ЭВМ и ПО, так и проектировщиков сетей ЭВМ. Если первых интересует влияние тех или иных структурных или функциональных изменений в ЭВМ на ее общую эффективность и производительность, то вторые рассматривают ЭВМ в контексте ее взаимодействия с другими компонентами сети. При этом для проектировщиков сетей важны

времена выполнения задач на клиентах (терминалах и сетевых рабочих станциях) и серверах, а также потоки связующей информации между ними. Модели ЭВМ для первой и второй групп пользователей предлагается создавать на одинаковых принципах. Различия появляются на этапе планирования и постановки имитационных экспериментов и заключаются методике проведения исследований.



а



б

Рис.1 Внешний вид интерфейса программы просмотра результатов мониторинга вычислительного процесса (*а* – компактный вид представления; *б* – подробный вид представления)

Принципы формализации персонального компьютера (ПК):

1). Представление вычислительного процесса в ПК с высокой степенью детализации, при этом сеть представляется как элемент ПК (внешнее устройство).

2). Декомпозиция ПК на компоненты, соответствующие его аппаратным частям.

ПК представляется состоящим из процессора, жесткого диска, видеоконтроллера, сетевого адаптера, ОЗУ. Программное обеспечение ПК находит свое отражение в алгоритмах функционирования компонент. Такой подход с одной стороны, упрощает декомпозицию объекта моделирования, а с другой стороны, позволяет реализовать неразрывную связь аппаратуры и программного обеспечения (ПО).

3). Объединение компонент ПК на основе процессов ОС.

4).Отражение в модели внутреннего параллелизма в ПК.

Производительность ЭВМ определяется в значительной мере степенью распараллеливания процессов обработки информации внутри нее.

В модели должна быть учтена возможность указанного распараллеливания процессов. Например, для имитации буфера записей процессор представляется в модели двумя обслуживающими устройствами: собственно процессором и шинным интерфейсом. «Интеллектуальность» ПУ и возможность прямого доступа закладывается в алгоритмы функционирования устройств, соответствующих ПУ. При этом имитируются и ограничения на параллелизм со стороны, используемой общей шины, а также общей логики задачи.

5). Моделирование вычислительного процесса в ЭВМ на конкретных функциональных задачах.

Функционирование ПК неразрывно связано с характером выполняемых на нем функциональных задач. Моделирование вычислительного процесса в ПК также выполняется на конкретных функциональных задачах. Функциональная задача (ФЗ) параметризуется, причем существует связь между параметрами ФЗ и параметрами самого ПК. Такое положение является следствием тесной связи между характеристиками аппаратуры и ПО. Так, например, если процессором используются пакетные циклы считывания из ОЗУ, то время считывания из ОЗУ увеличивается, а частота обращений к памяти сокращается. Этот факт находит отражение в значениях параметра ПК «время считывания из ОЗУ» и параметра ФЗ «средний интервал между запросами процессора на считывание из памяти».

Модель ФЗ можно отнести к параметрическим моделям, когда ФЗ представляются без учета семантики их алгоритмов в виде смеси команд характеризующейся теми же параметрами, что и реальные ФЗ. Причем, упор делается на характеристики ФЗ, порождающие процессы обмена информацией внутри ПК. ФЗ при этом представляется как совокупность случайных потоков запросов на обмен с памятью и ПУ, ограниченных во времени числом команд. Для ФЗ, обрабатываемой на терминале, рассматриваются потоки запросов: на запись в память; на чтение из ОЗУ внешней кэш-памяти; на чтение или запись на жесткий диск; к видеоконтроллеру; к сети; к пользователю. Для ФЗ сервера и сетевой рабочей станции рассматриваются те же потоки, с единственным уточнением, что параллельно обмену с сетью они могут выполнять внутренние задачи.

Для каждого терминала или сервера выделяется несколько типов ФЗ, которые могут выполняться на нем. Типы ФЗ идентичны по составу параметров, но отличаются их значениями. Статистики моделирования собираются отдельно по каждому типу ФЗ.

6). Представление времени выполнения ФЗ на ПК как сложной функции от расчетных параметров и результатов моделирования.

Время выполнения ФЗ на ПК определяется временем обслуживания ее на процессоре. В рамках длины программы в командах процессор в модели выполняет команды по одной, имитируя выполнение каждой команды задержкой на среднее время выполнения команды. При этом на каждой команде он проверяет наличие и при необходимости выполняет запросы на взаимодействие с памятью (запись, считывание), диском, дисплеем, сетью, пользователем. При выполнении этих запросов процессор пользуется общей шиной ПК, передавая по ней информацию и инициализируя работу соответствующих компонент ПК. Если шина занята (идет, например, передача ответа, из сети в память), то процессор должен ожидать ее освобождения. Кроме того, процессор ожидает окончания выполнения некоторых запросов (например, запросов к диску на чтение, запросов к сети). Таким образом, время выполнения ФЗ складывается из времени выполнения самой программы на процессоре и времени ожидания процессора. Если время выполнения может быть вычислено аналитически перемножением, длины программы в командах на среднее время выполнения команды, то время ожидания (t_w) зависит от операционной обстановки в ПК и складывается из времен ожидания освобождения шины (t_{wbus}), окончания обмена по шине (t_{wintf}) и окончания выполнения запросов (t_{fques}):

$$t_w = t_{wbus} + t_{wintf} + t_{fques}.$$

Таким образом, время выполнения ФЗ является сложной функцией и носит случайный характер. Для определения его среднего знамения и поиска закона распределения более всего подходит метод статистических испытаний.

7).Применение аналитических расчетов для уменьшения числа событий в модели.

Число событий в модели напрямую влияет на время постановки имитационного эксперимента. Поэтому следует стремиться к уменьшению числа событий, если это не затрагивает основную концепцию модели. Например, при моделировании диска с внутренним буфером предлагается не имитировать подробно параллельные процессы обмена данных между буфером диска и магнитной поверхностью и обмена данными между буфером диска и

ОЗУ. Вместо этого отрабатываются интервалы занятия общей шины ПК в интересах диска по оригинальному алгоритму с использованием аналитического расчета длительности этих интервалов.

8). Моделирование квазипараллельной обработки функциональных задач на сервере или сетевой рабочей станции с одним процессором.

Предлагаются две схемы организации квазипараллелизма обработки ФЗ на сервере:

а) о переключением на другую ФЗ при возникновении состояния ожидания завершения чтения о диска;

б) о переключением на другую ФЗ через заданный интервал времени.

Реализация различных схем организации квазипараллелизма дает возможность сравнить эффективность схем на конкретной нагрузке на ПК.

Помимо информации, необходимой для реорганизации вычислительных сетей данная система также позволит существенным образом снизить деформацию вычислительного процесса на любом узле или персональном компьютере, не входящем в состав сети, путем исследования его рабочей нагрузки, анализе узких мест и выдачи рекомендаций по адаптации аппаратной базы.

Литература

1. Ресурсы Microsoft Windows NT Workstation 4.0 BHV Санкт-Петербург, 1998.
2. Грек В.В., Максимей И.В. Стандартизация и метрология систем обработки данных: Учеб. пособие. Мн.: Выш. шк., 1994. 287 с.
3. Еськова О.И. Разработка метода имитационного моделирования сетей обработки данных. Автореферат диссертации, ГГУ им.Ф. Скорины, Гомель, 1996. 22 с.

ON A TECHNOLOGY MONITORING AND NATURE EXAMINATION OF A LOCAL NETWORK

A.V. Varuev, O.M. Demidenko, V.A. Nikishaev, M.V. Potrashkova

Annotation

The actuality of a local network examinate under the project modeling process of the organizational variants of a computing process has been shown. Monitoring of a computing process for nature examination systems on workstation local network for tests it influence to traffic local network and search recommendations to remuve this situation is wrote. Information from the monitor use as input information for the imitation modeling. The major parameters and imitational modeling responses of the organisational variants of a computing process the working load have been defined. The sample of system for visualization date recive from monitor is showed.