
Модернизация унаследованных СУБД для обеспечения параллелизма

Самарев Роман Станиславович
samarev@acm.org

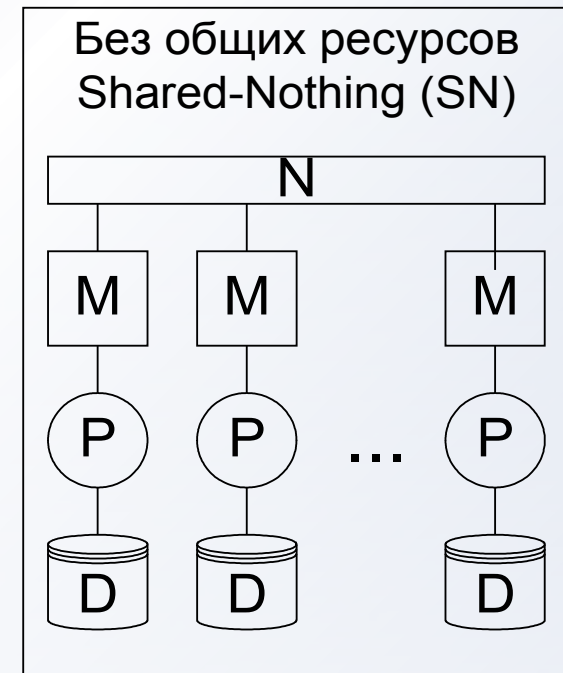
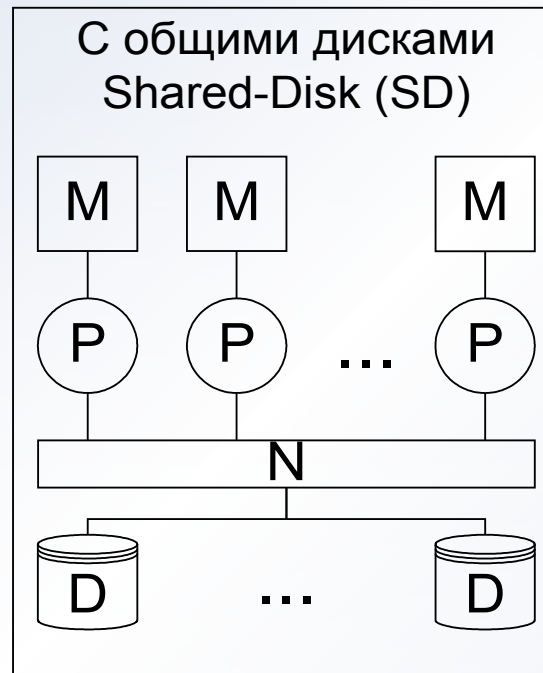
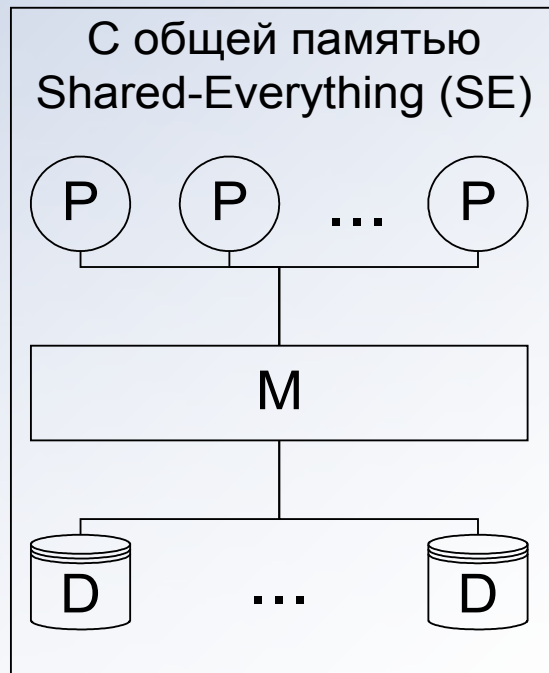
Актуальность параллельного ПО

- Масштабирования ВС и увеличение производительности
- Повышение эффективности использования оборудования и балансирование нагрузки
 - Минимизация времени отклика
- Повышение устойчивости к сбоям

Основы параллельных СУБД

Типы параллельных ВС для СУБД

Классификация по Стоунбрейкеру



Р – процессор

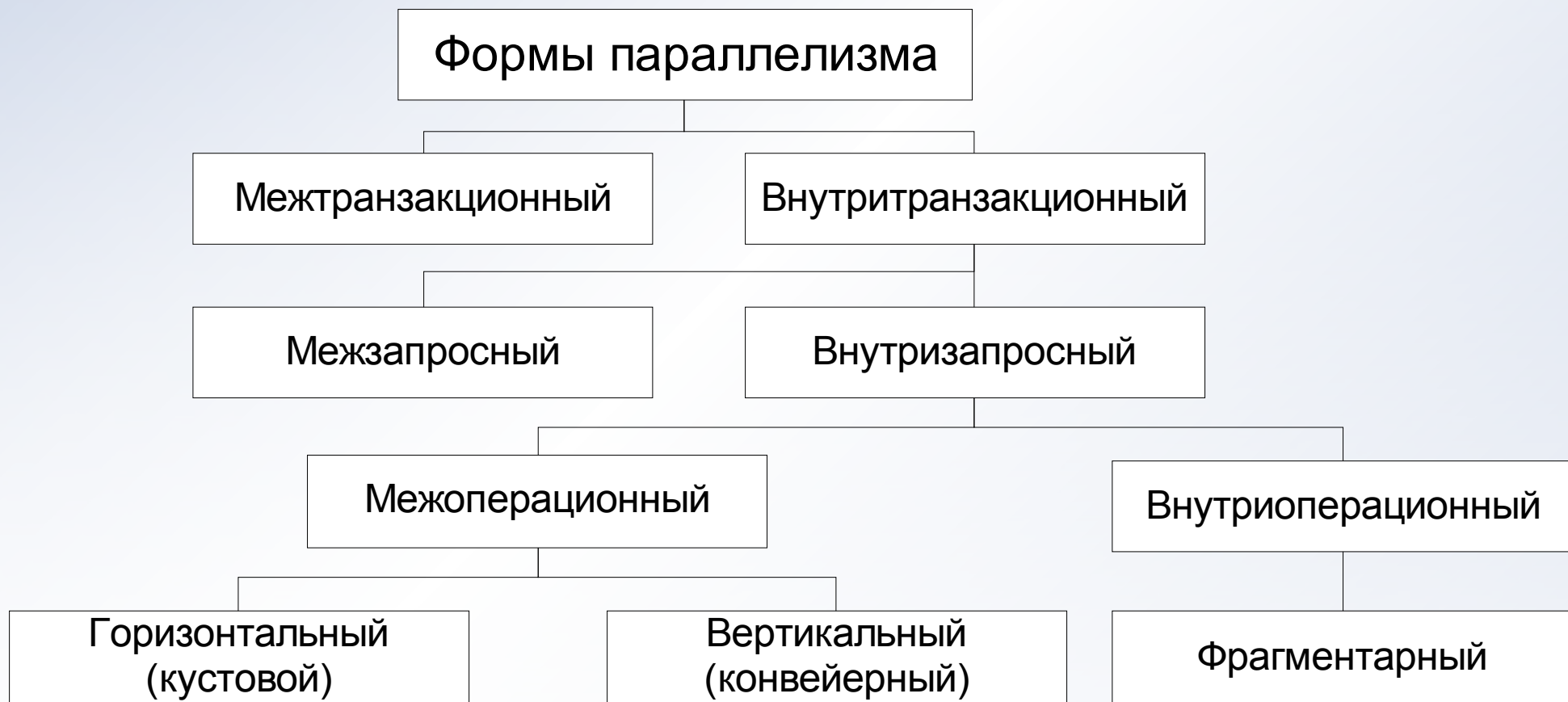
М – оперативная память

Д - дисковый накопитель

Н – коммуникационная сеть

Формы параллелизма

(Graefe G, Соколинский Л.Б.)



Не конвейерный параллелизм

Горизонтальный и фрагментарный параллелизм

ОQL-запрос

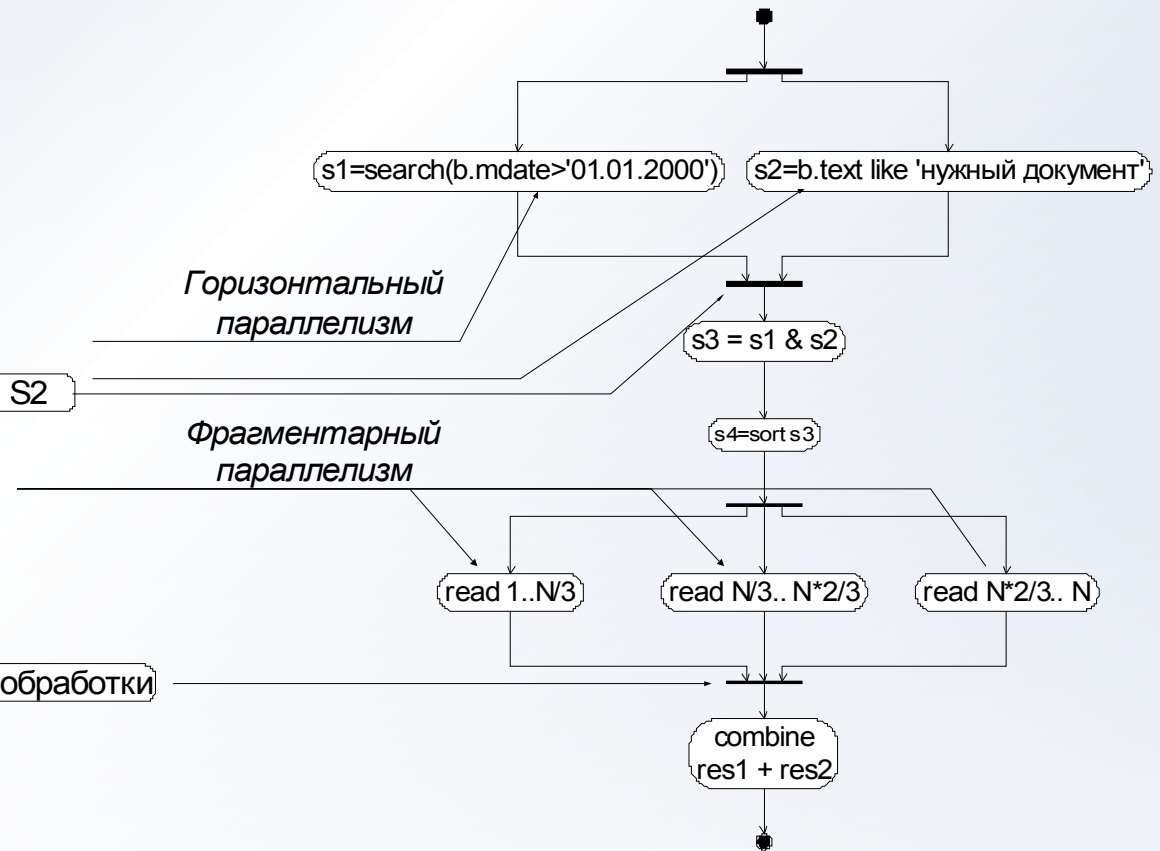
```
select b.name, b.id, b.ref_rubric.name  
from Document b  
where b.mdate > '01.01.2000' and  
       b.text like 'нужный документ'  
order by b.mdate
```

Физический план выполнения

```
s1 = search(b.mdate > '01.01.2000');  
s2 = b.text like 'нужный документ';  
s3 = s1 & s2;  
s4 = sort s3 by b.mdate;  
N = s4.size();  
for( i=0; i<N; i++)  
{  
  res1 = add ( read ( s4[i], b.name, b.id, b.ref_rubric ) );  
  res2 = add ( read ( b.ref_rubric.oid, b.ref_rubric.name ) );  
}
```

```
result = res1 + res2;
```

Алгоритм выполнения запроса

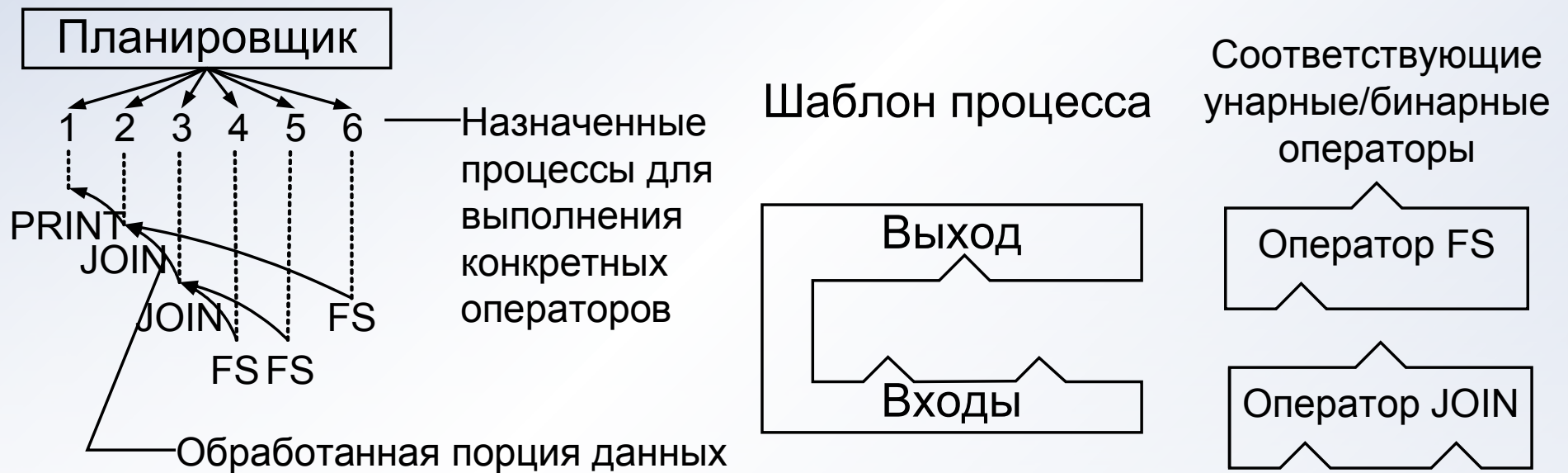


Ожидание S1 и S2

Ожидание окончания обработки

Конвейерный параллелизм

скобочный параллелизм



Конвейерный параллелизм

операторный параллелизм

Физический план
выполнения запроса

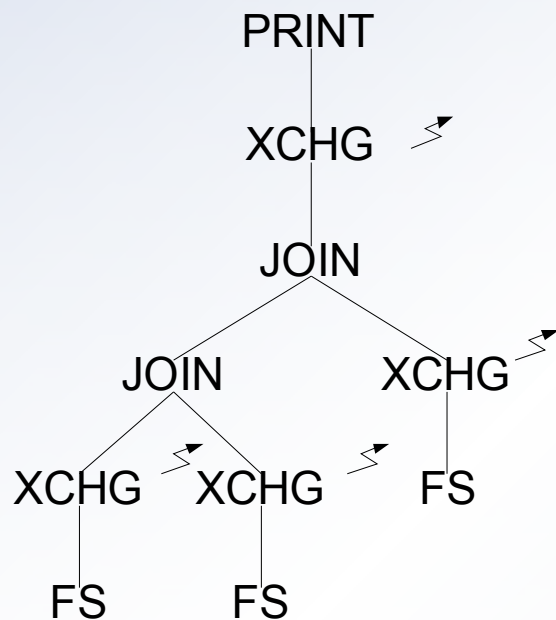
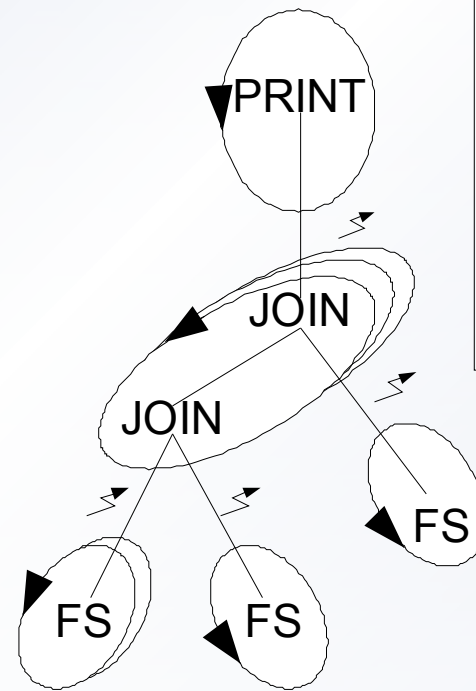


Схема возможного
распределения процессов



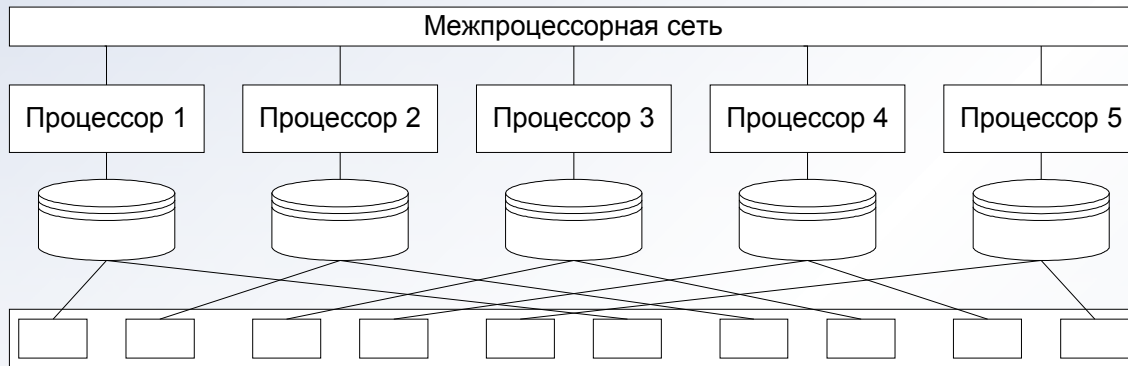
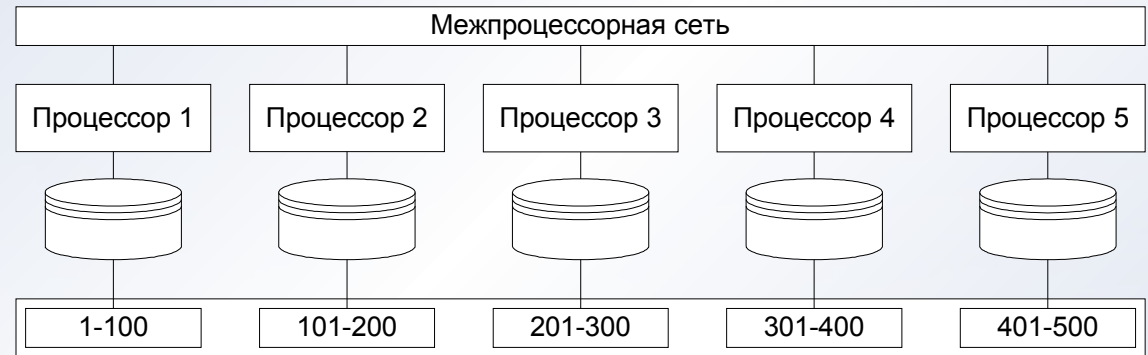
Планировщик

Выбор или создание
процесса для
выполнения и
передача порции
данных

Отдельный
циклический
процесс в
конвейере

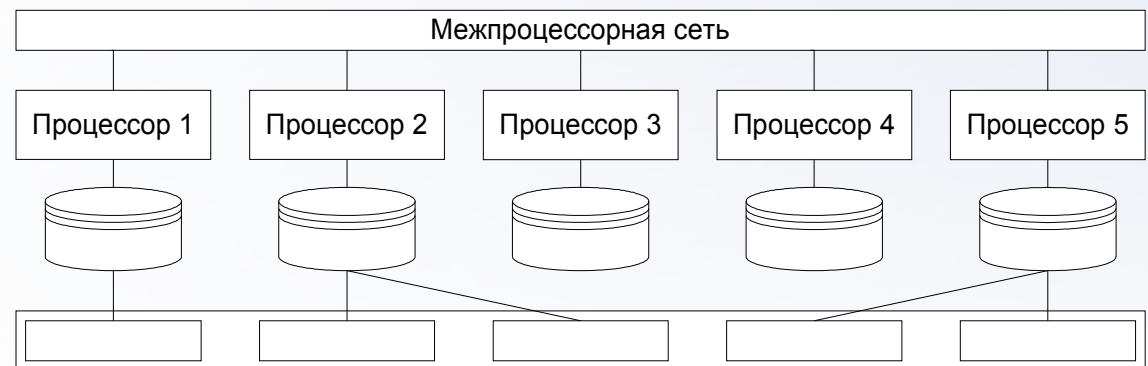
Фрагментация данных

Фрагментация по диапазонам



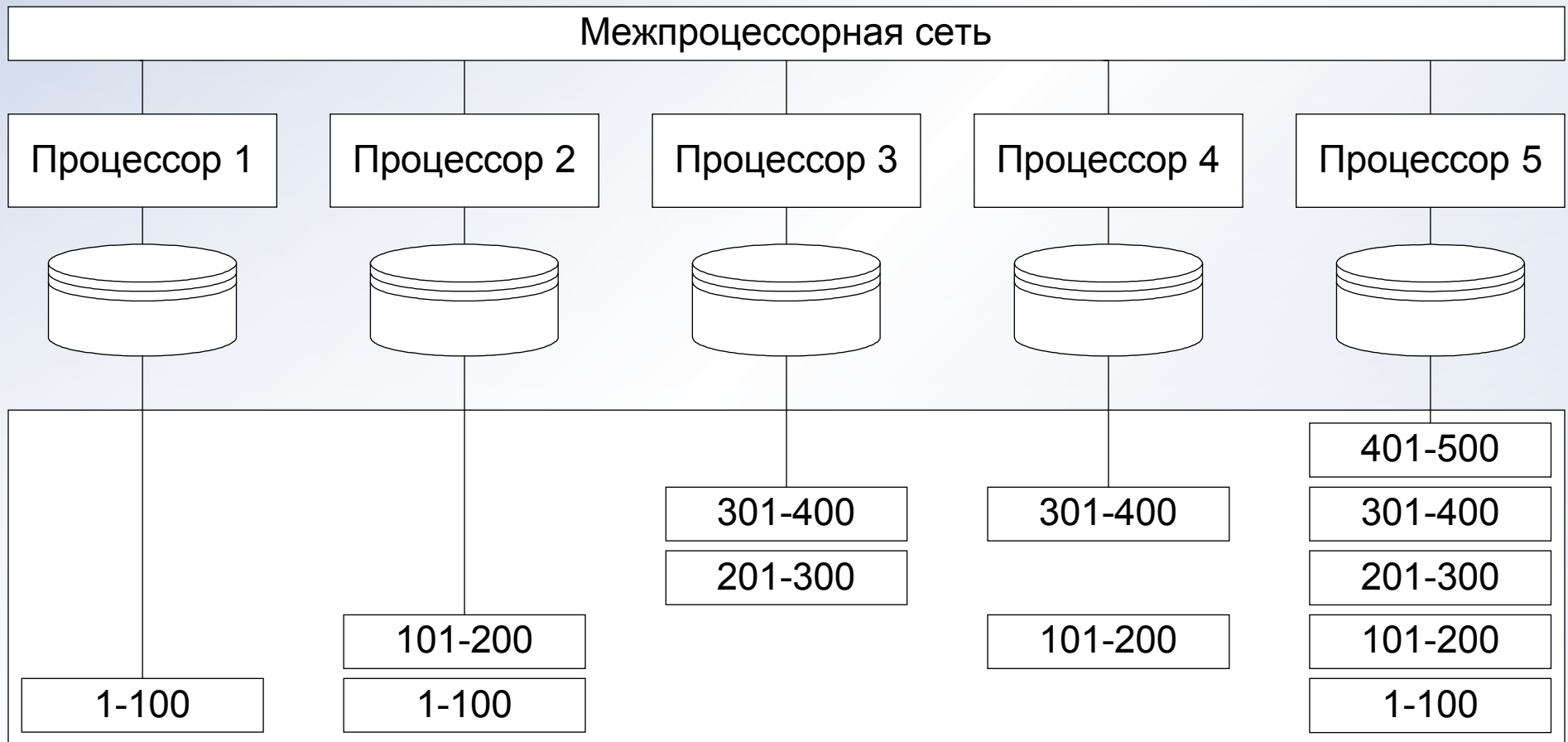
Карусельная фрагментация

Фрагментация хэшированием



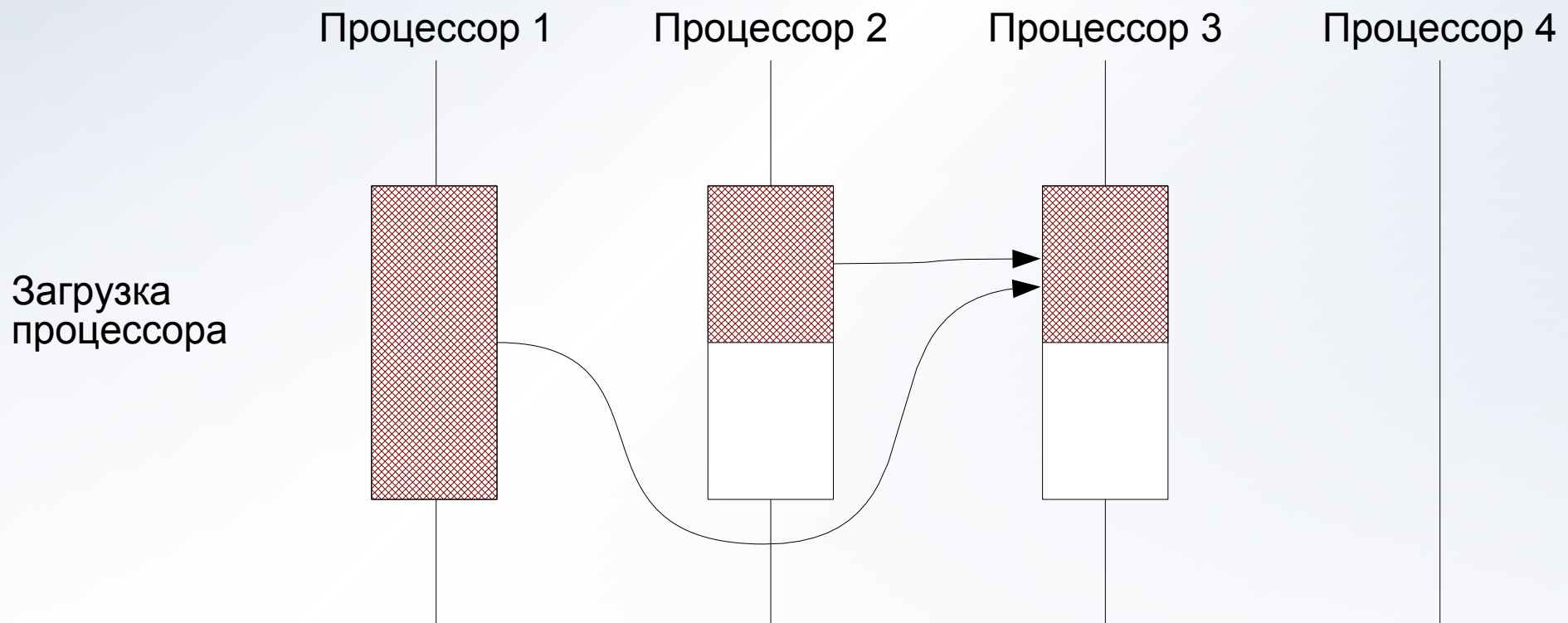
Фрагментация данных

Фрагментация с избыточностью данных



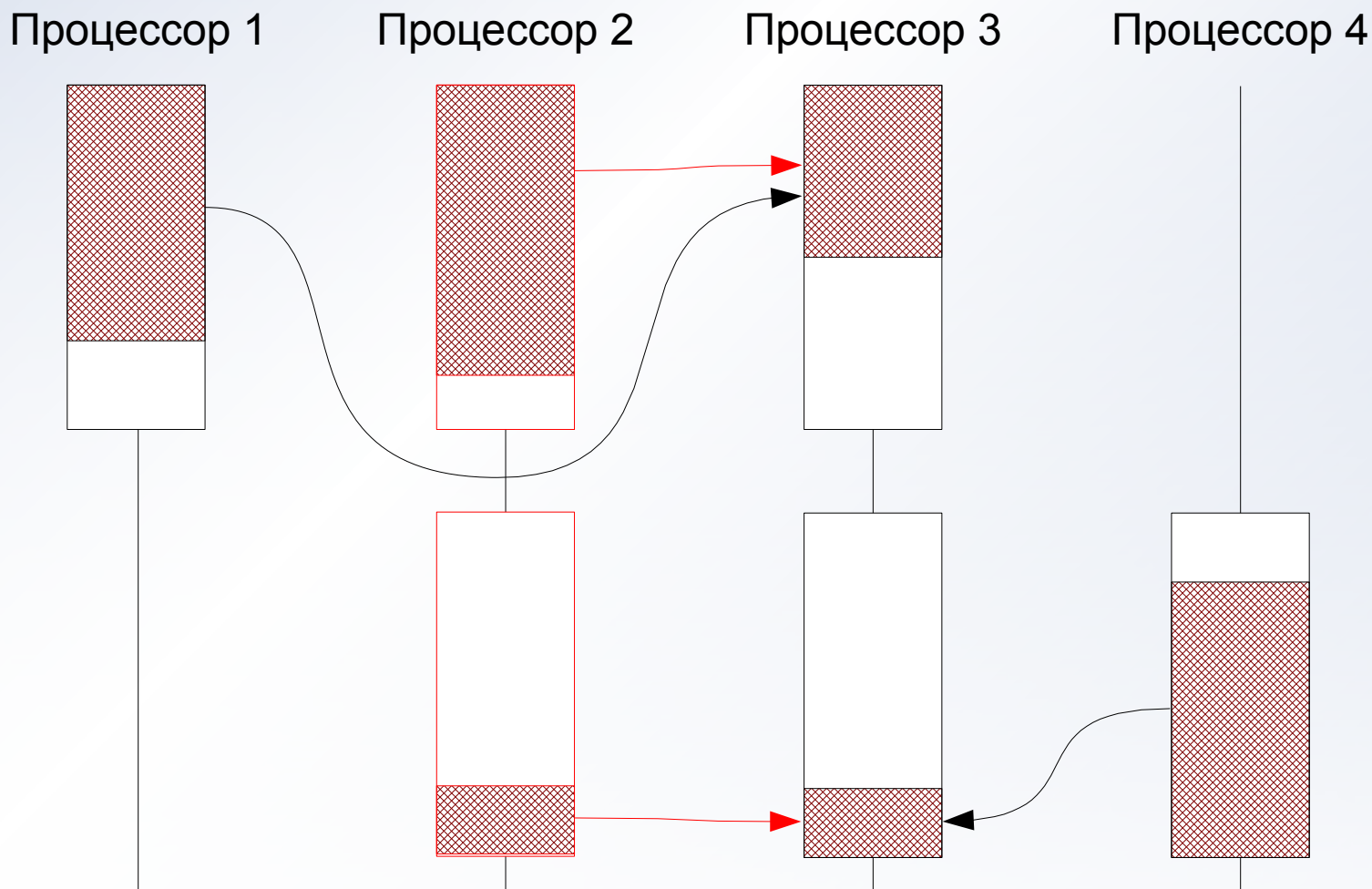
Балансирование нагрузки процессоров

Переком вследствие ошибки оценки или недостаточного объема данных



Балансирование нагрузки процессоров

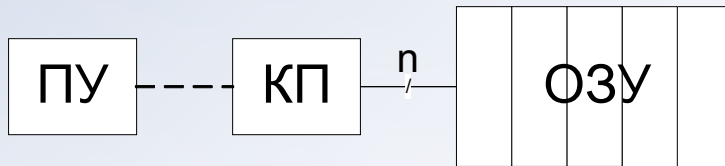
Перегрузка отдельных процессоров



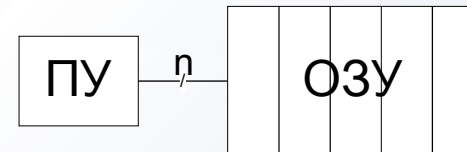
Различия архитектуры ВС в рамках группы SE

Подсистемы памяти ВС семейства x86 (фрагментация памяти)

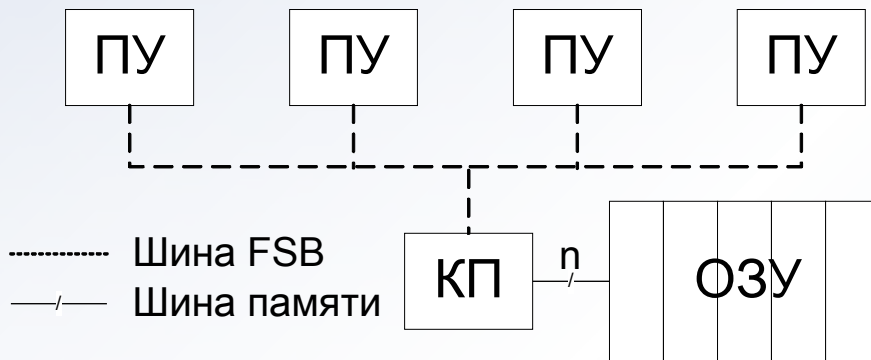
Intel Pentium 4



AMD Athlon 64

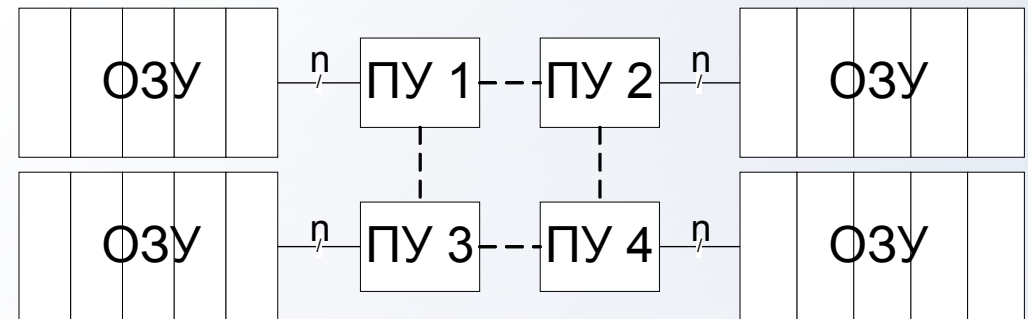


Intel Xeon



ПУ - процессорный узел
КП - контроллер памяти
n - количество каналов памяти

AMD Opteron (4 ПУ)



----- Шина HyperTransport
—/— Шина памяти

Основные направления параллельной обработки

Выборка данных

- Фрагментация данных
- Создание вспомогательных структур для промежуточного хранения (кэширование, избыточное кэширование)

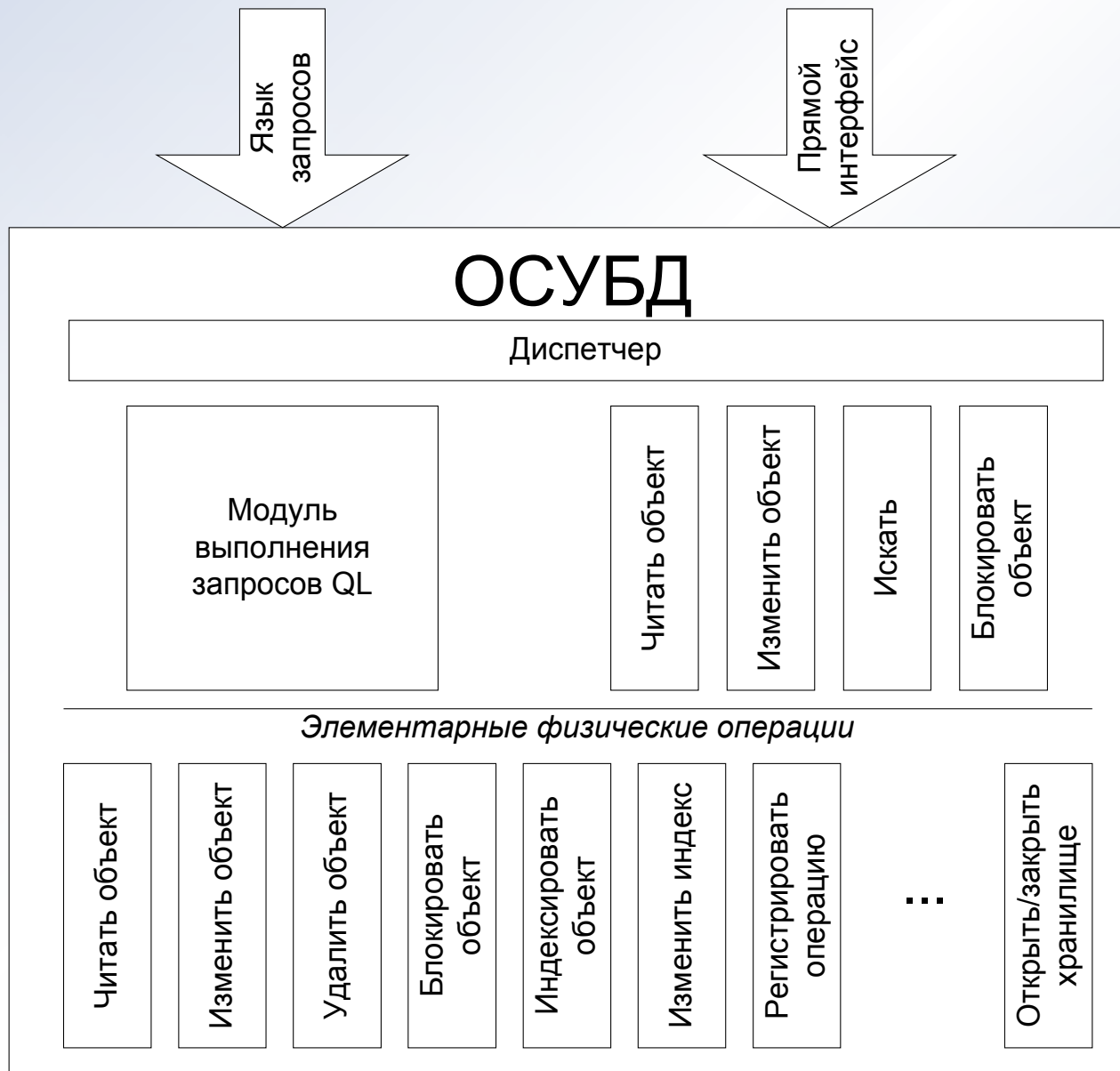
Модификация данных

- Транзакции с упорядоченными режимами блокировки данных
- Фрагментация данных
- Минимизация размеров внутренних блокировок
 - Специальные индексные структуры и методы записи данных на жесткий диск

Параллельный план выполнения

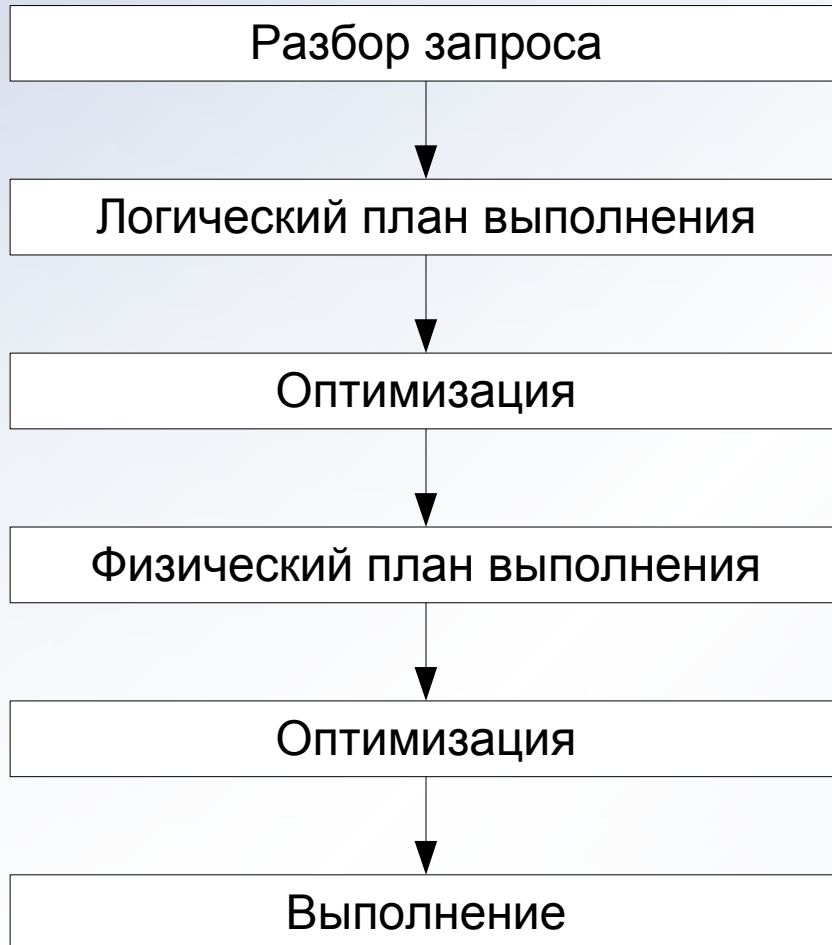
Интерфейсы СУБД

Запрос для ОСУБД

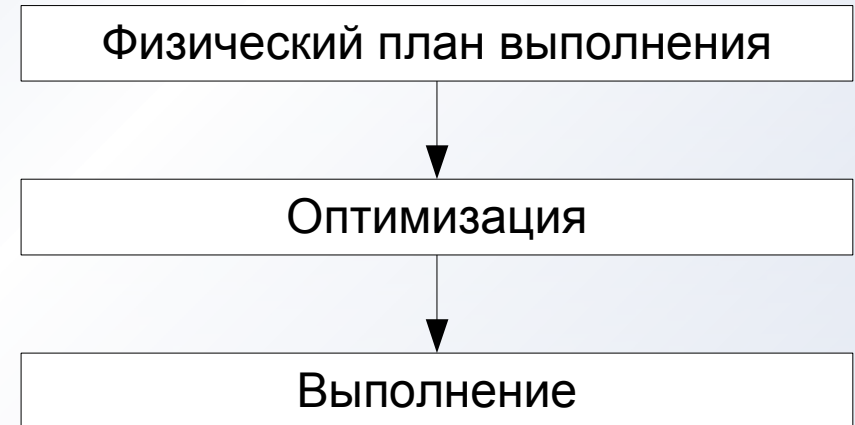


Этапы преобразования запроса

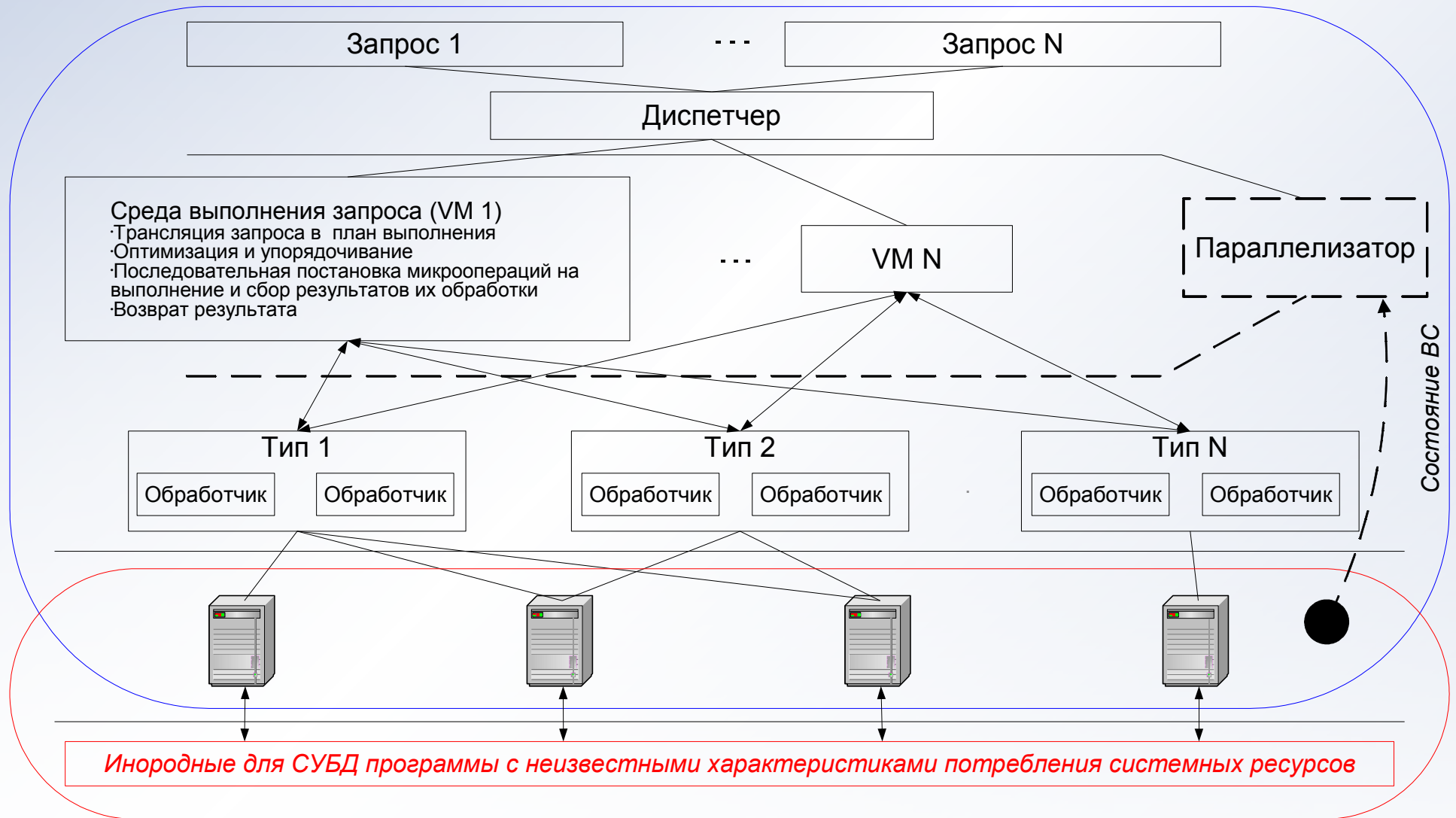
на языке запросов



интерфейс прямого доступа



Обобщенная схема обработки запросов



Создание параллельного плана выполнения

Планирование и выполнение в статическом,
жестко установленном режиме

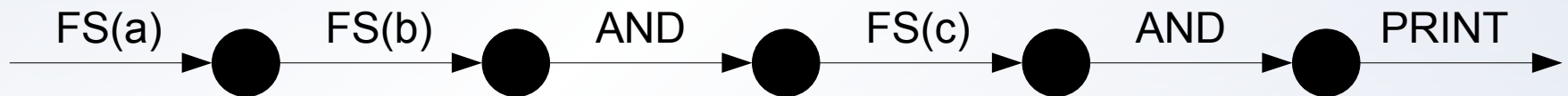
Планирование с максимальной степенью
параллелизма и выполнение по текущему
состоянию СУБД

Анализ последовательности выполнения запроса

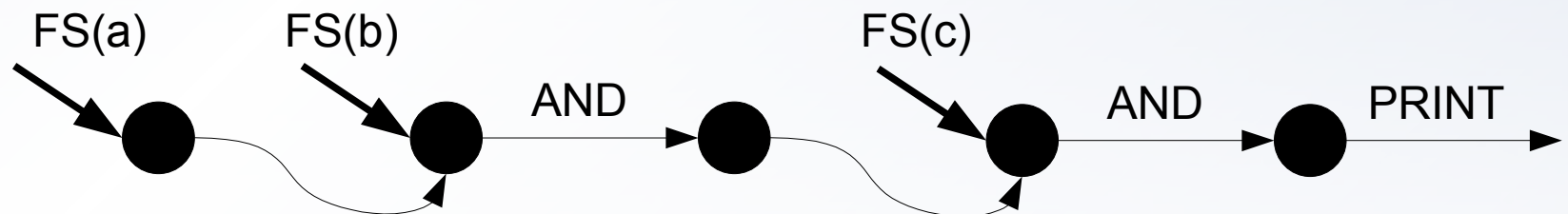
Запрос

```
SELECT a FROM db WHERE a=1 AND b=2 AND c=3;
```

Последовательное выполнение



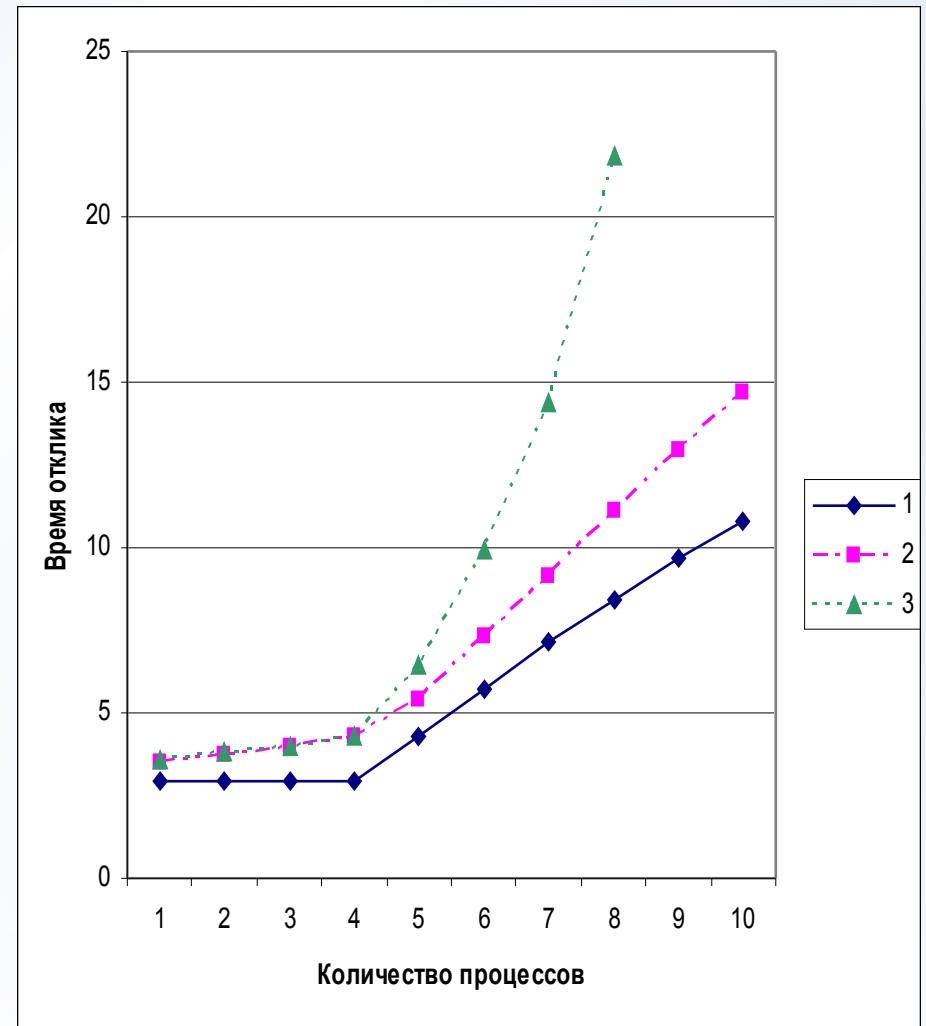
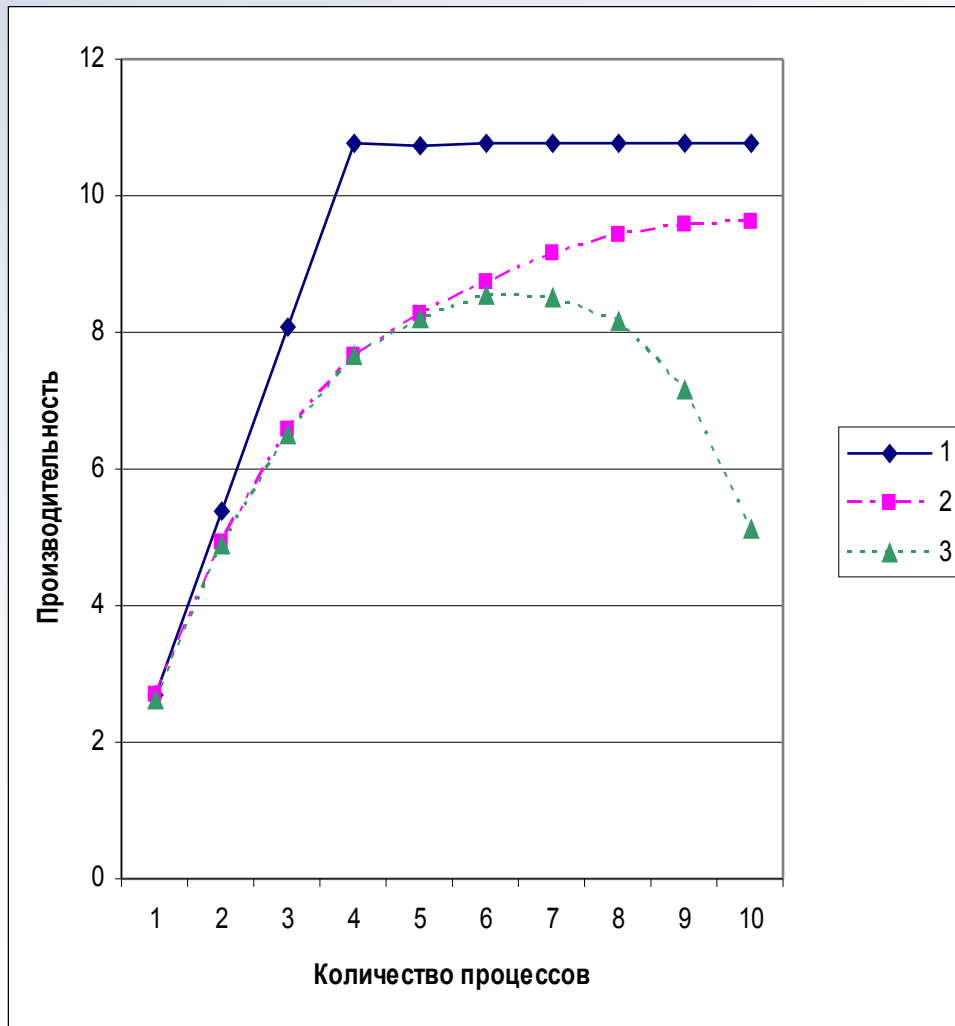
Граф зависимости



Производительность системы при параллельном выполнении операций

Характеристики:

- 1 - теоретическая, 2 – реальная с ограничением и очередью,
- 3 – реальная без ограничения (деградация производительности)



Проблема совместного использования ресурсов

Пример: одновременное выполнение операций оп. 1, 2, 3 в условиях совместного использования ресурсов

	ОЗУ, МБ	HDD, R/W, сектор	ЦП, мс
Оп.1	600	W 1, 100, 101	200
Оп.2	300	R 2, 3, 4, 200	100
Оп.3	500	R 50, 51, 52, 1	150
Доступно ВС	1024	1 HDD	2 ЦП
Следствие одновременного выполнения	Страничный обмен	Ожидание позиционирования головок	Увеличение частоты переключения контекста процессов
Оценка времени выполнения	-	-	$>t_1+t_2+t_3$

Итоговое время выполнения – не определимо

Способы организации параллельного выполнения программ

Основные средства организации параллельного выполнения

- MPI

Обмен сообщениями между процессами. Функции Send,Recv. Является библиотекой функций.

- OpenMP

Распараллеливание участков кода программы, фрагментарный параллелизм циклов посредством разметки кода специальными директивами. Требует поддержки компилятором.

- Системные средства

Средства управления процессами/потоками, предоставляемые операционной системой.

Общий недостаток –

необходимость ручного управления параллелизмом

Системные средства параллельного выполнения

- Средства запуска потоков (нитей) / процессов
- Средства синхронизации
- Средства оценки эффективности средств синхронизации

MS Windows

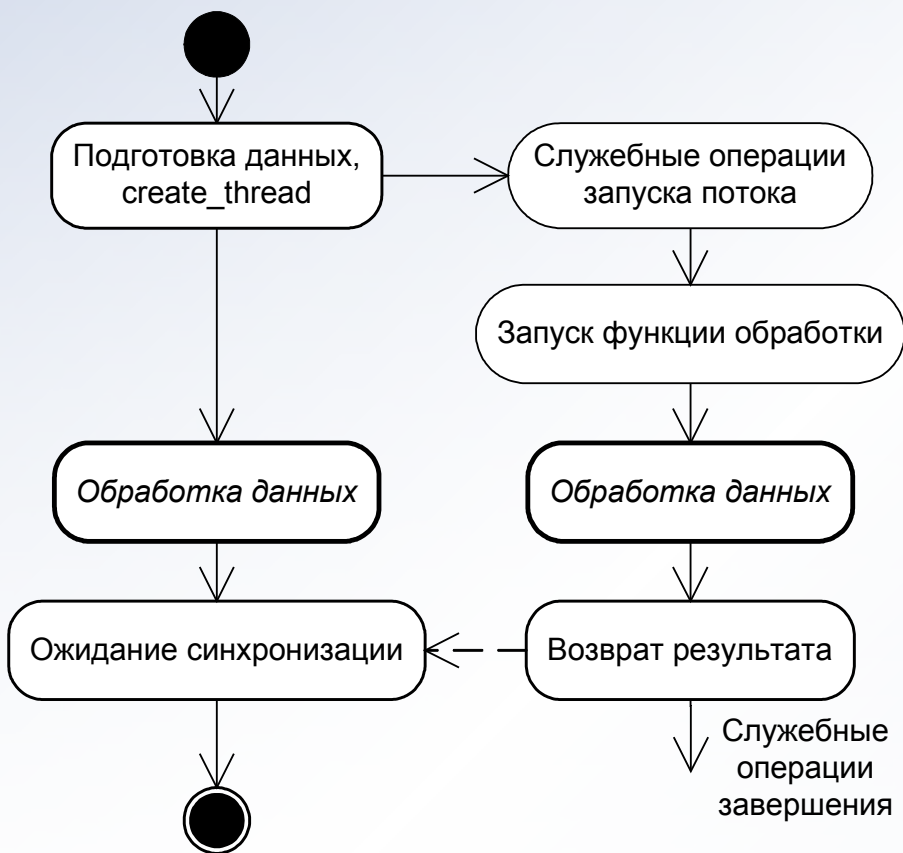
- `create_thread` / `Resume-/SuspendThread` / `Process` / `Fiber`
- `CriticalSection` / `Event` / `Semaphore` / `Mutex`
- `GetTickCount` / `rdtsc`

Linux

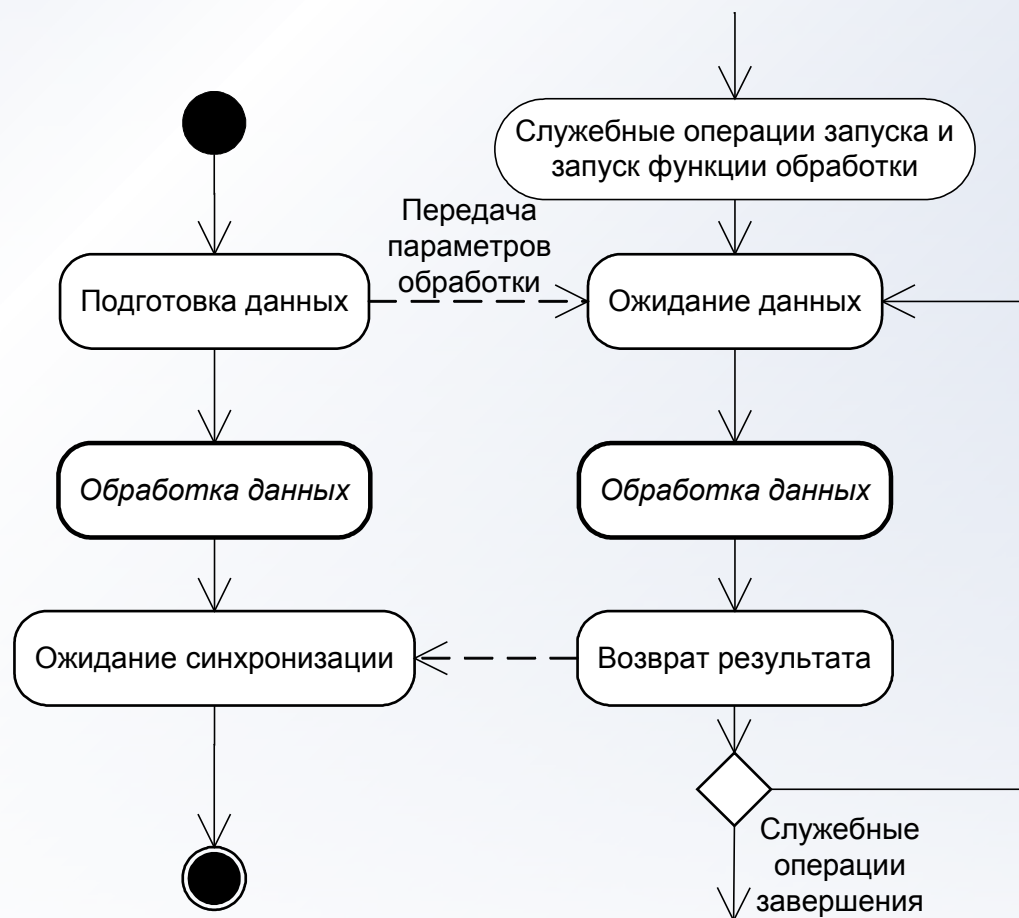
- `pthread` / `signal` / `fork`
- `pthread_mutex`
- `rdtsc`

Выбор способа запуска параллельной обработки

Непосредственный запуск потока перед обработкой



Предварительный запуск потока до обработки



Выбор способа запуска параллельной обработки

Примеры способов организации временной приостановки потоков (упрощенно)

Рабочий поток

```
...  
do {  
    while( !bStartCmd )  
        Sleep(1);  
  
    bStartCmd = false;  
  
    ...  
    // полезная работа  
    ...  
  
} while( !bExit );  
...
```

Управляющий поток

```
set_data(...);  
bStartCmd = true;  
wait_result(...);
```

Рабочий поток

```
...  
  
SuspendThread(WorkThr);  
  
do {  
    ...  
    // полезная работа  
    ...  
  
    SuspendThread(WorkThr);  
} while( !bExit );  
...
```

Управляющий поток

```
set_data(...);  
ResumeThread(WorkThr);  
wait_result(...);
```

Рабочий поток

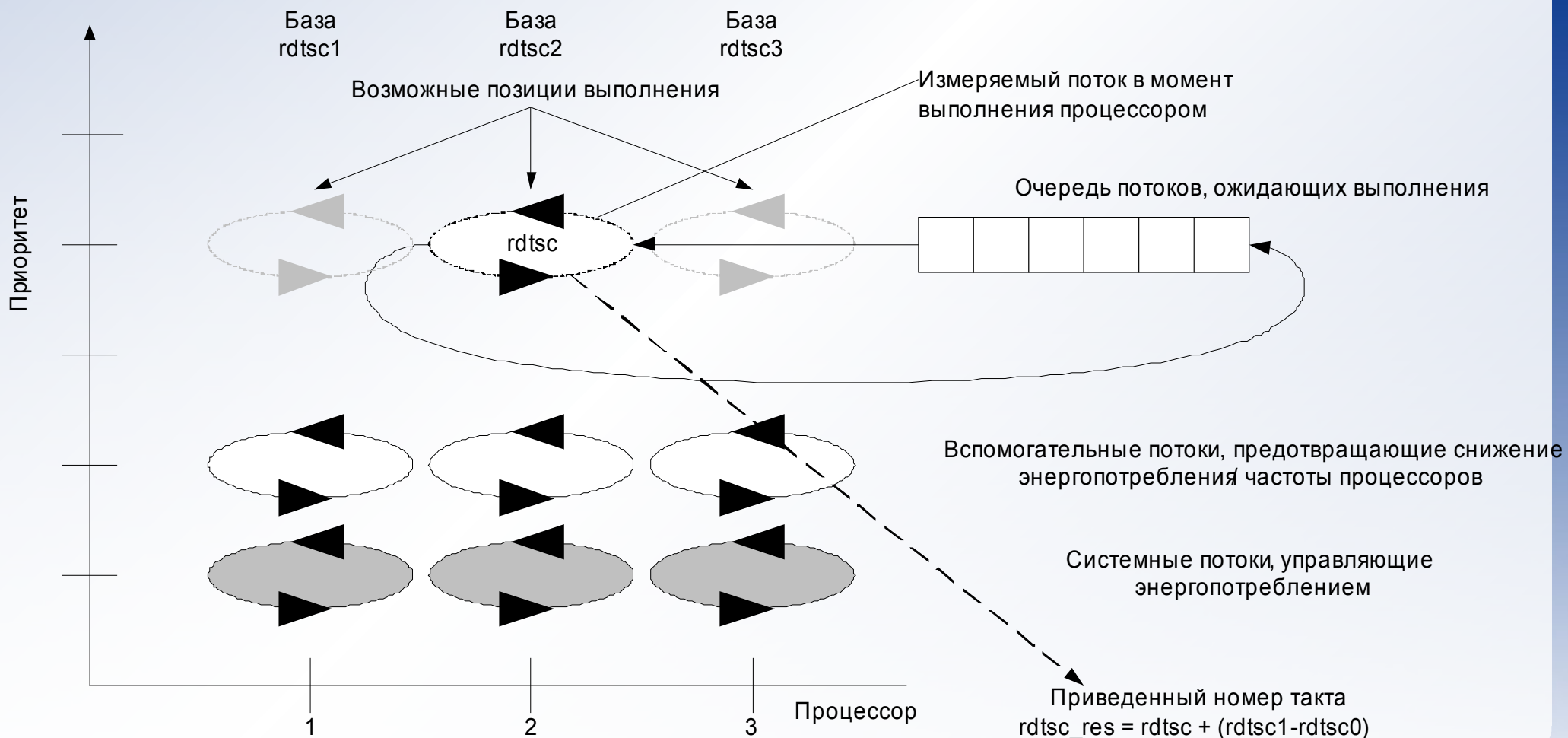
```
...  
do {  
    WaitForSingleObject(  
        Ev, INFINITE);  
  
    ResetEvent(Ev);  
  
    ...  
    // полезная работа  
    ...  
  
} while( !bExit );  
...
```

Управляющий поток

```
set_data(...);  
SetEvent( Ev );  
wait_result(...);
```

Выбор способа запуска параллельной обработки

Оценка малых интервалов времени с помощью счетчика тактов **rdtsc**



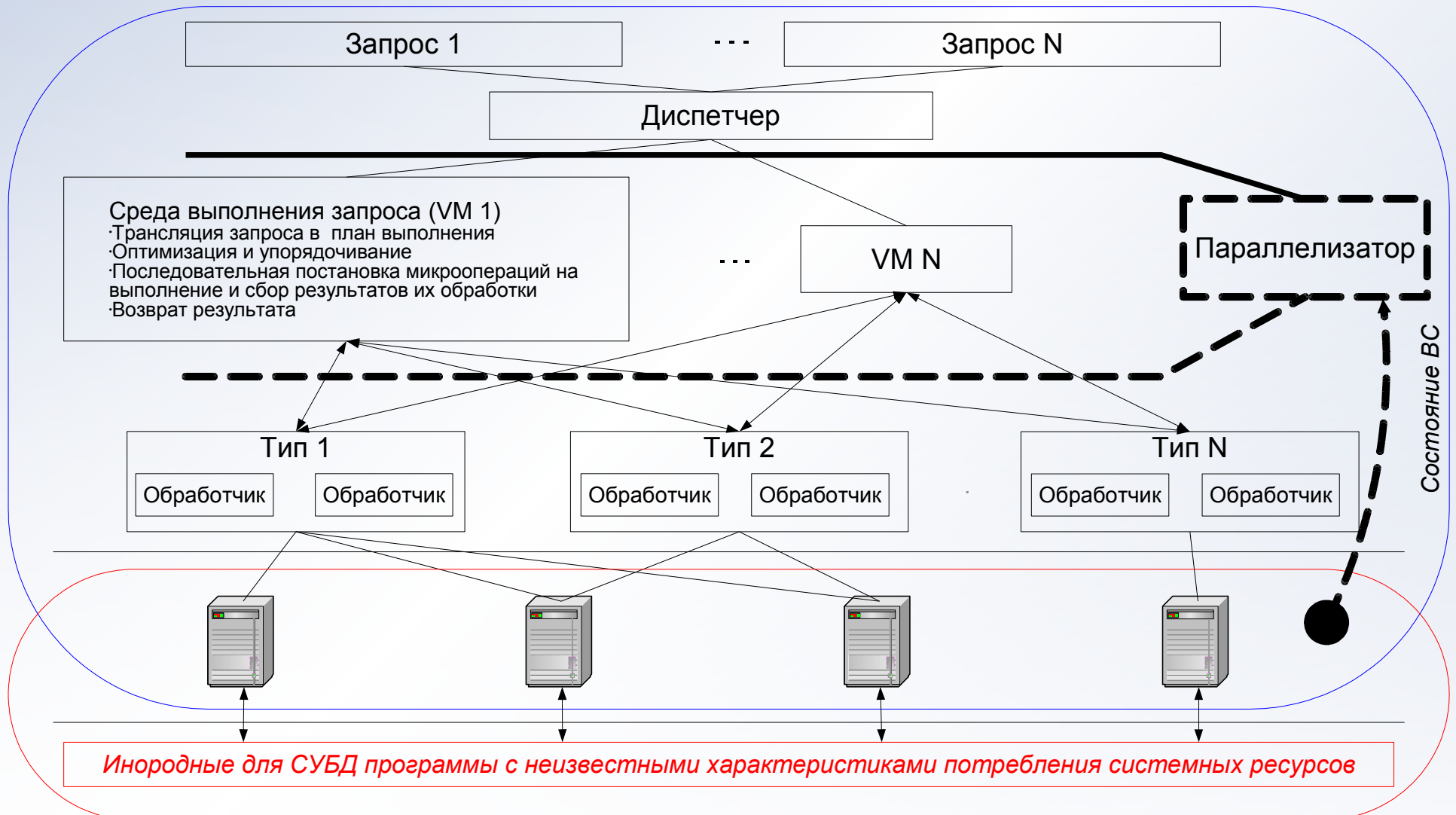
Выбор способа приостановки параллельных потоков

Результаты оценки времени выполнения операций с использованием Windows API

Операция	Процессор, время выполнения операции, мкс				
	AMD Turion ML-34 Windows XP SP2	AMD Athlon 64 X2 4400+ Windows XP SP2	Intel Pentium 4 3.0 S775 Windows 2003 Server	Intel Pentium 4 2.8 S478 Windows XP SP2	*Intel Xeon 4 2.6 * 4 шт Windows 2000 Server
Выполнение Sleep(1) – потеря кванта времени процессора	35035	15616	1956	1571	15718
Создание нового потока _createthread	161	135	92	115	266
Запуск потока ResumeThread	25	81	15	34	155
Синхронизация событиями SetEvent-WaitForSingleObject	22	19	16	16	79
Синхронизация критическими секциями LeaveCriticalSection-EnterCriticalSection	2,8	2,6	6,6	4,43	20

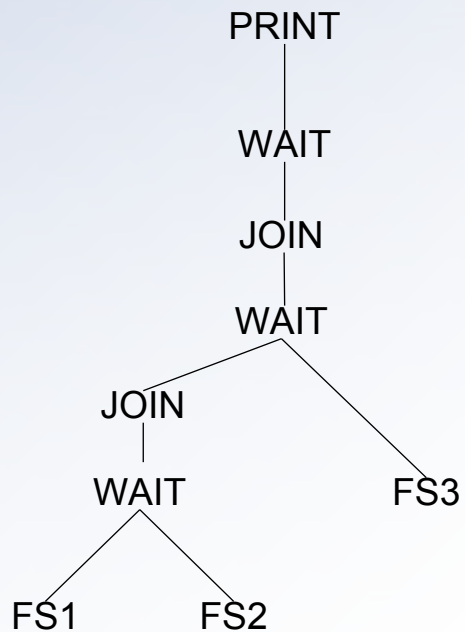
Разработка параллелизатора

Управление процессом параллельного выполнения



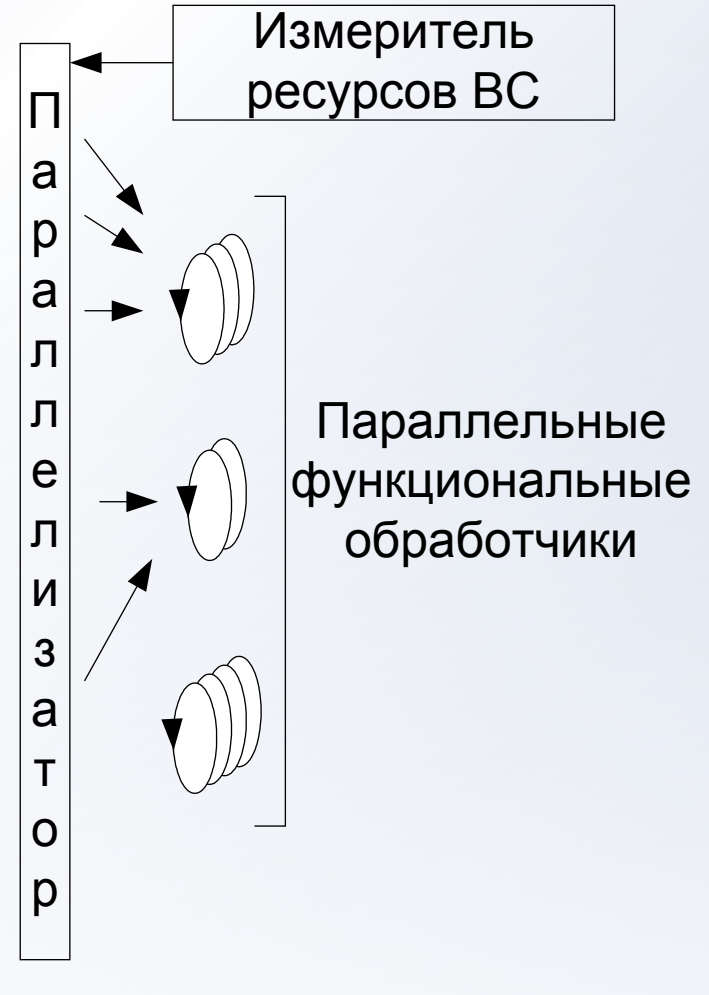
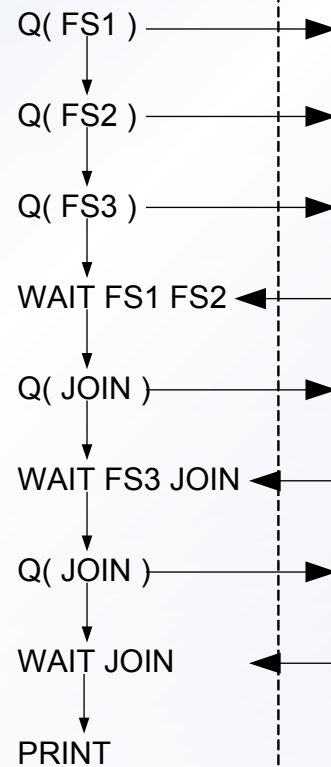
Физический план выполнения

Возможный параллельный физический план

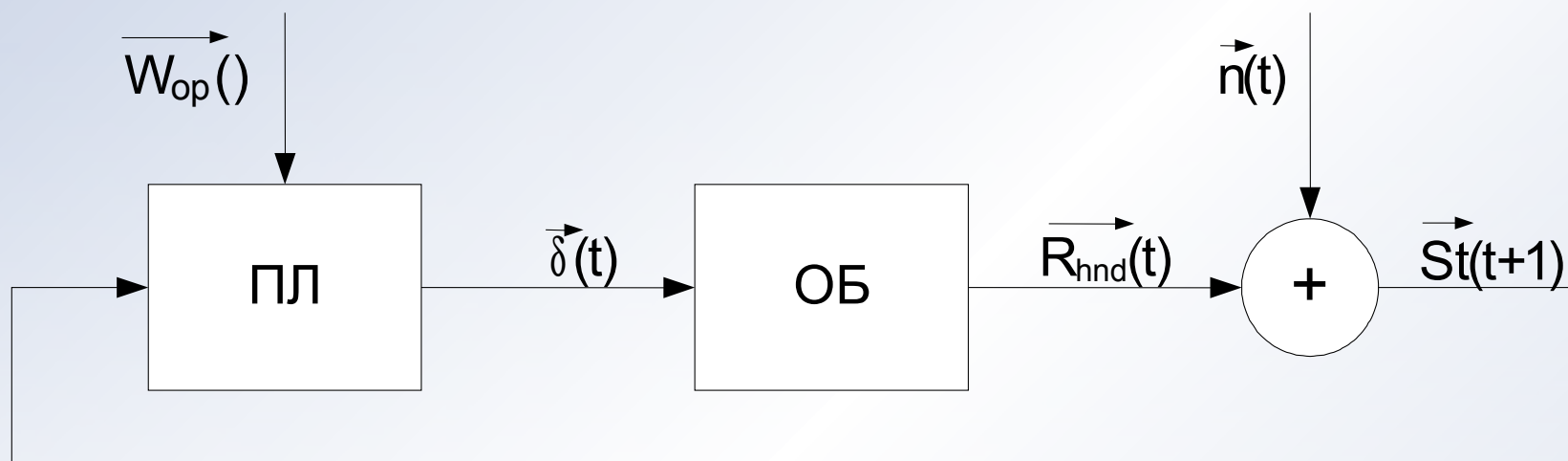


Управляющий

ПОТОК



Обобщенная схема управления параллелизмом



ПЛ

– планировщик

ОБ

– объект управления (обработчики)

$\vec{W}_{op}(t)$

– априорный вес операции

$\vec{\delta}(t)$?

– вектор изменения состояния обработчиков

$\vec{R}_{hnd}(t)$

– вектор фактически потребленных ресурсов

$\vec{n}(t)$

– вектор использования системных ресурсов
внешними программами

$\vec{S}t(t+1)$

– вектор состояния ВС

Создание обратной связи по состоянию ВС

MS Windows

Счетчики производительности (Performance Counters)

- **Параметры процессоров** - процент времени простоя, длина очередей операций, количество операций в секунду.
- **Параметры подсистемы памяти** - объем, интенсивность обращения по операциям чтения/записи, параметры подсистем кэширования и страничного обмена, объемы доступной и использованной памяти.
- **Параметры дисковой подсистемы** – относительные и абсолютные значения объемов передачи данных по операциям чтения/записи данных, объемы очередей операций по каждому физическому и логическому диску, процент времени простоя.
- **Общесистемные параметры** - количество файловых операций и системных вызовов, длина очередей процессоров, количество процессов и потоков в системе.
- **Параметры специального назначения** – формируются конкретными приложениями, работающими на данной ВС.

Обработка значений счетчиков производительности

$$\vec{n}'(t) = F_{PC}(\vec{PC})$$

\vec{PC} - вектор, образованный мгновенными значениями счетчиков производительности

$\vec{n}'(t)$ - вектор состояния вычислительной системы

Порядок анализа счетчиков производительности

1. Определение размерности вектора состояния ВС
2. Определение размерности вектора мгновенных значений счетчиков производительности:
 - Экспертная фильтрация счетчиков производительности (СП)
 - Анализ информативности СП для идентификации необходимых режимов ВС
3. Определение функции преобразования

Формальное описание параллелизатора

Параллелизатор: $Par = \langle Hnd, W_{SYS}, W_{cur} \rangle$, W_{SYS}, W_{cur} – векторы доступных ресурсов и текущего состояния системы;

Множество обработчиков: $Hnd = \{hnd\}$, где $hnd_j = \langle \{op\}, st_j \rangle$, $op \in A$, A – множество типов операций;

Текущее состояние j-го обработчика $st_j = \langle n_c, N_c, I_{in}, I_{out}, L_{in}, L_{out} \rangle$, n_c, N_c – число активных и доступных каналов обработки, I_{in}, L_{in} – текущая и допустимая длины входной очереди, I_{out}, L_{out} – текущая и допустимая длины выходной очереди;

Операция $op_i = \langle \{t_{in}\}, \{t_{out}\}, W_{op,i} \rangle$, где $t_{in}, t_{out} \in T$ – типы входных и выходных параметров, T – множество типов данных;

Заявка $q = \langle No, pr, params:op_i, W_q \rangle$, где No – последовательный номер операции в системе, pr – приоритет заявки;

$params:op_i = \langle \{param_{in}:t_{in}\}, \{param_{out}:t_{out}\} \rangle$ – множества значений входных и выходных параметров соответствующих типов;

W_{op}, W_q – априорный вес операции, вес операции на основании оценки сложности выполнения

Формальное описание параллелизатора

$$W_{cur} = \begin{bmatrix} rs_1 \\ rs_2 \\ \dots \\ rs_N \end{bmatrix}$$

Вектор текущего состояния
 rs_j - величина потребления j -го ресурса
 N – число типов ресурсов

$$W_{SYS} = \begin{bmatrix} RS_1 \\ RS_2 \\ \dots \\ RS_N \end{bmatrix}$$

Вектор доступных ресурсов
(частный случай)
 RS_j – Допустимая величина потребления j -го ресурса, полученная на основе априорных данных или выполненного тестирования

$$W_{op,i} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_N \end{bmatrix}$$

Вес операции
 w_j – величина потребления j -го ресурса, полученная на основе априорных данных или в результате накопленных статистических данных во время работы системы
 i – тип операции

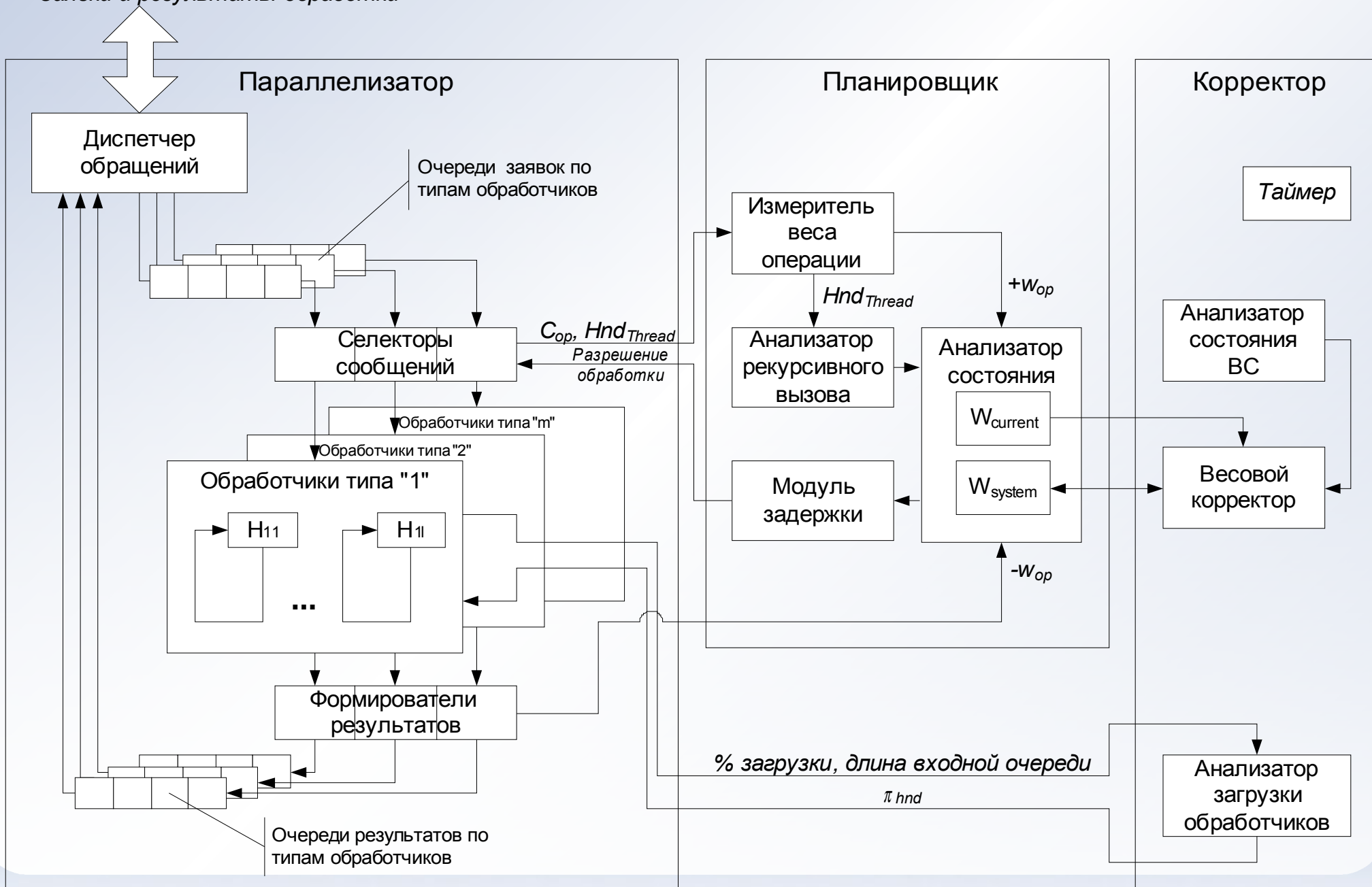
Условие постановки заявки на обработку

Частный случай $W_{SYS}^j \geq W_{cur}^j + W_{op,i}^j$
 j – номер ресурса

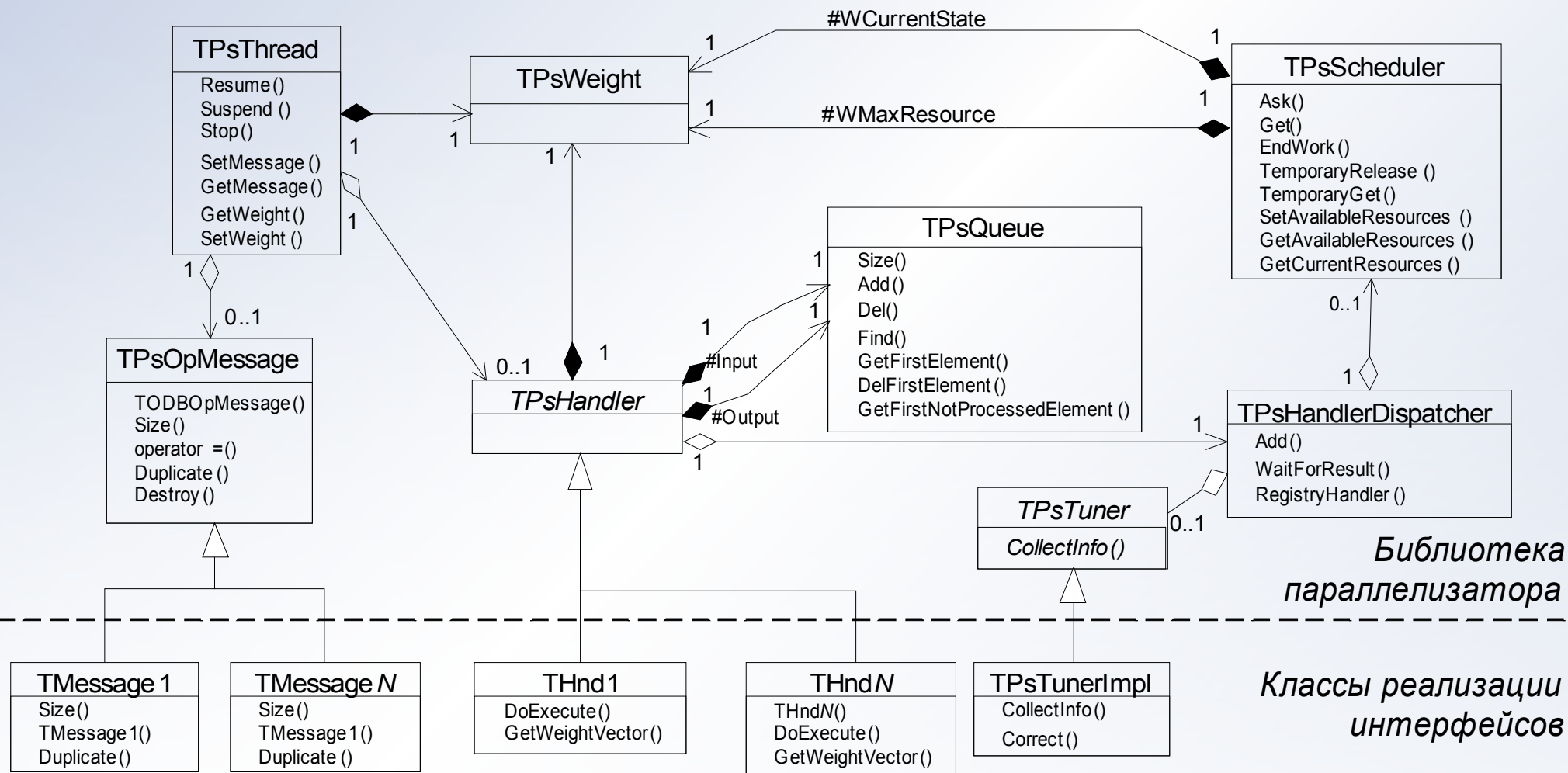
Общий случай $\|W_{SYS}^* (W_{cur}^1 + W_{op,i}^1, \dots, W_{cur}^N + W_{op,i}^N)\| \leq 1$
 $W_{SYS}^*(\dots)$ – функция, описывающая гиперповерхность комбинаций допустимых ресурсов

Адаптивный параллелизатор с асинхронным корректором

Заявки и результаты обработки

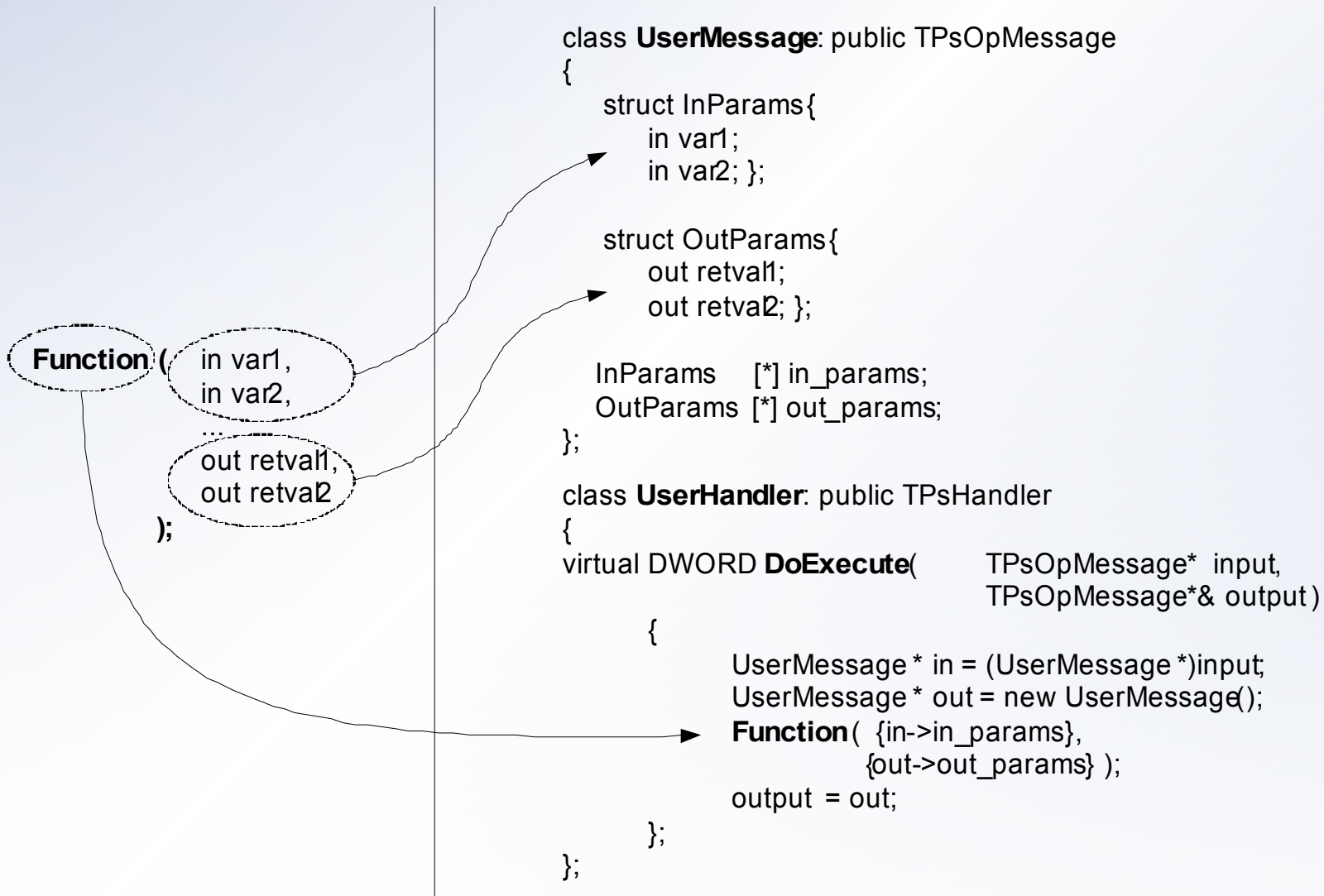


Программный интерфейс параллелизатора



Программный интерфейс параллелизатора

Преобразование функций в программах

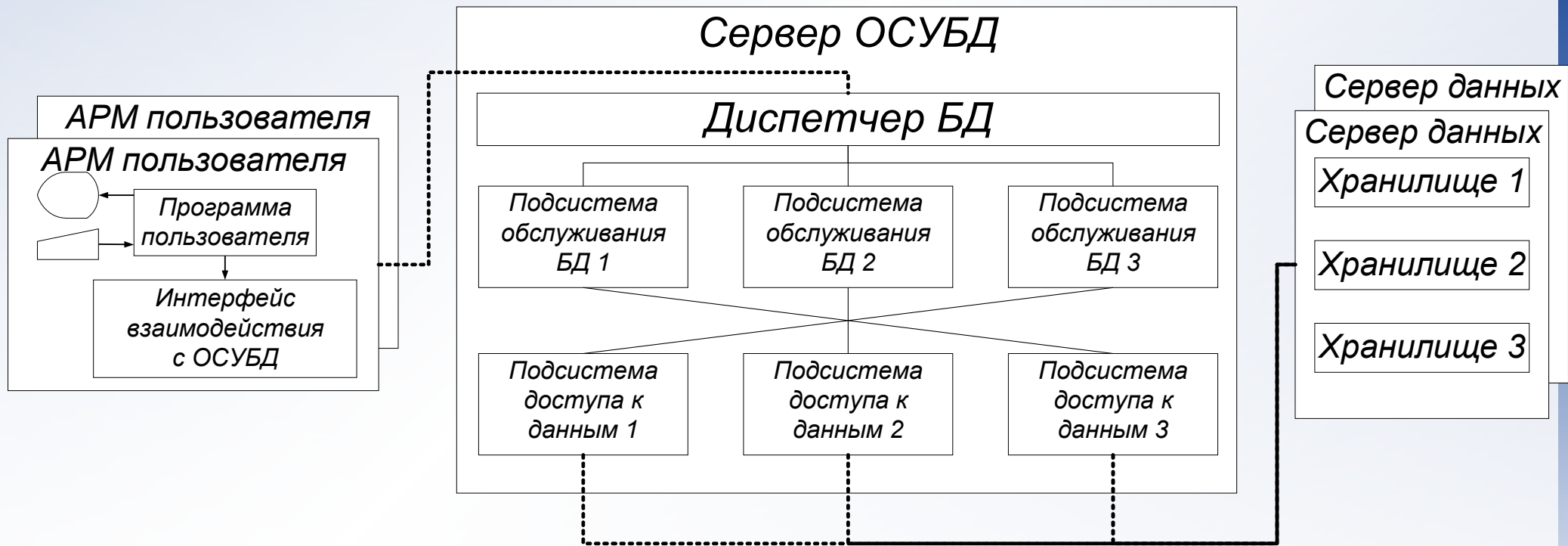


Модернизация СУБД для организации параллелизма на примере СУБД ODB-Jupiter

ОСУБД ODB-Jupiter

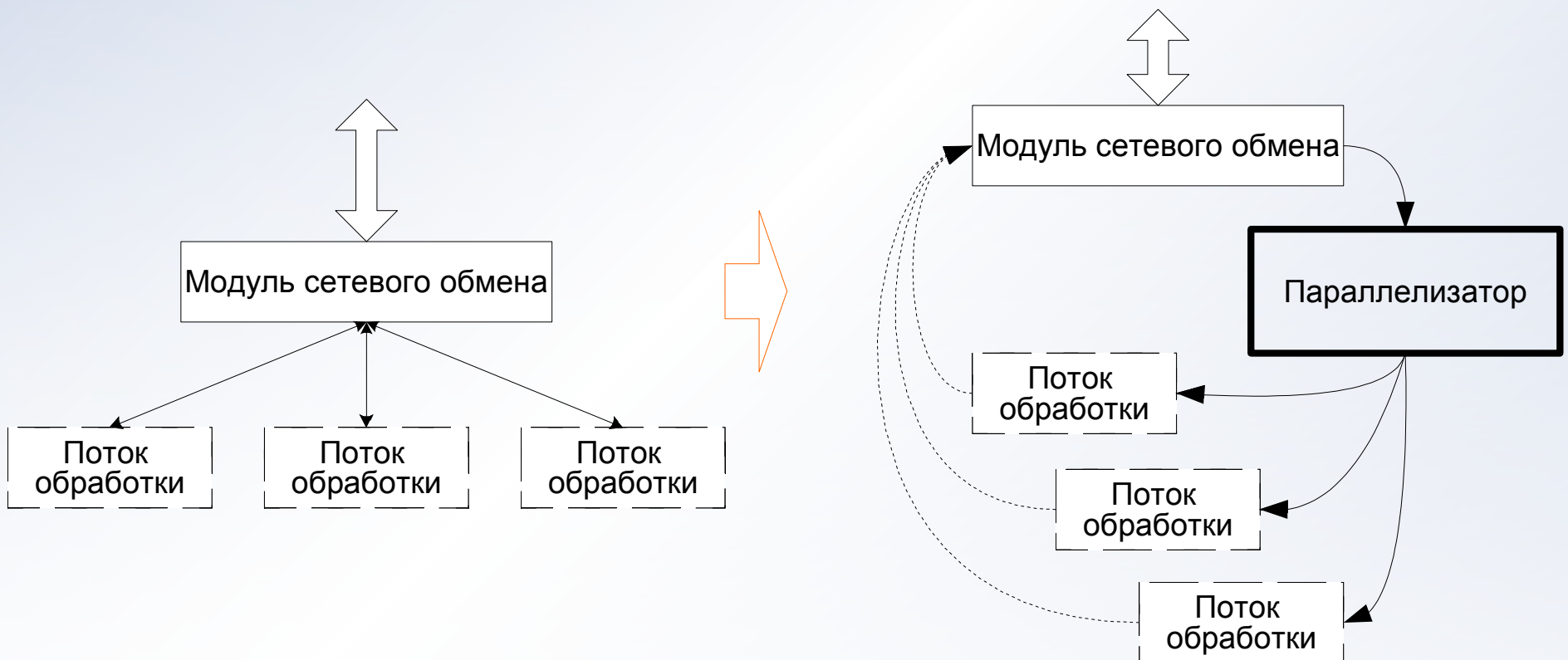
Общая схема организации системы ОСУБД
(Разработка НПЦ «ИНТЕЛТЕК ПЛЮС»)

<http://www.inteltec.ru>



Модификация существующих схем параллельной обработки

Унификация управления в межзапросных формах параллелизма



Выявление фрагментов кода, подлежащих распараллеливанию

Наименование операции	Комментарий	Тип операции на уровне п/с обслуж. БД	Тип операции на уровне п/с доступа к данным
Open	Открыть базу данных	НЭ	-
Close	Закрыть базу данных	НЭ	-
Read	Читать объект	НЭ	НЭ
Add	Добавить объект	НЭ	Э
Del	Удалить объект	НЭ	Э
Change	Изменить объект	НЭ	Э
Search	Поиск объектов	НЭ	НЭ
Lock	Блокировать/разблокировать объект	НЭ	НЭ
GetDataScheme	Получить схему данных	НЭ	-
CommitWork	Принять транзакцию	Э	Э
RollbackWork	Откатить транзакцию	Э	Э
BatchProcessing	Пакетная обработка запросов. В пакете могут содержаться операции Add, Del, Change, Search, Lock, CommitWork, передаваемые далее на обработку.	Зависит от содержимого пакета	-

*Примечание:

Э – эксклюзивная, НЭ – не эксклюзивная, '-' – нет операции

Выявление фрагментов кода, подлежащих распараллеливанию

Пример: операция модификации IntChange

Блокировать доступ к объекту в хранилище

Прочитать размер объекта из хранилища

Выделить область памяти для объекта

Прочитать объект из хранилища

Снять блокировку доступа к объекту в хранилище

Проверить новый объект и скорректировать старым

¹Заменить объект в хранилище данных

²Индексировать все реквизиты старого объекта

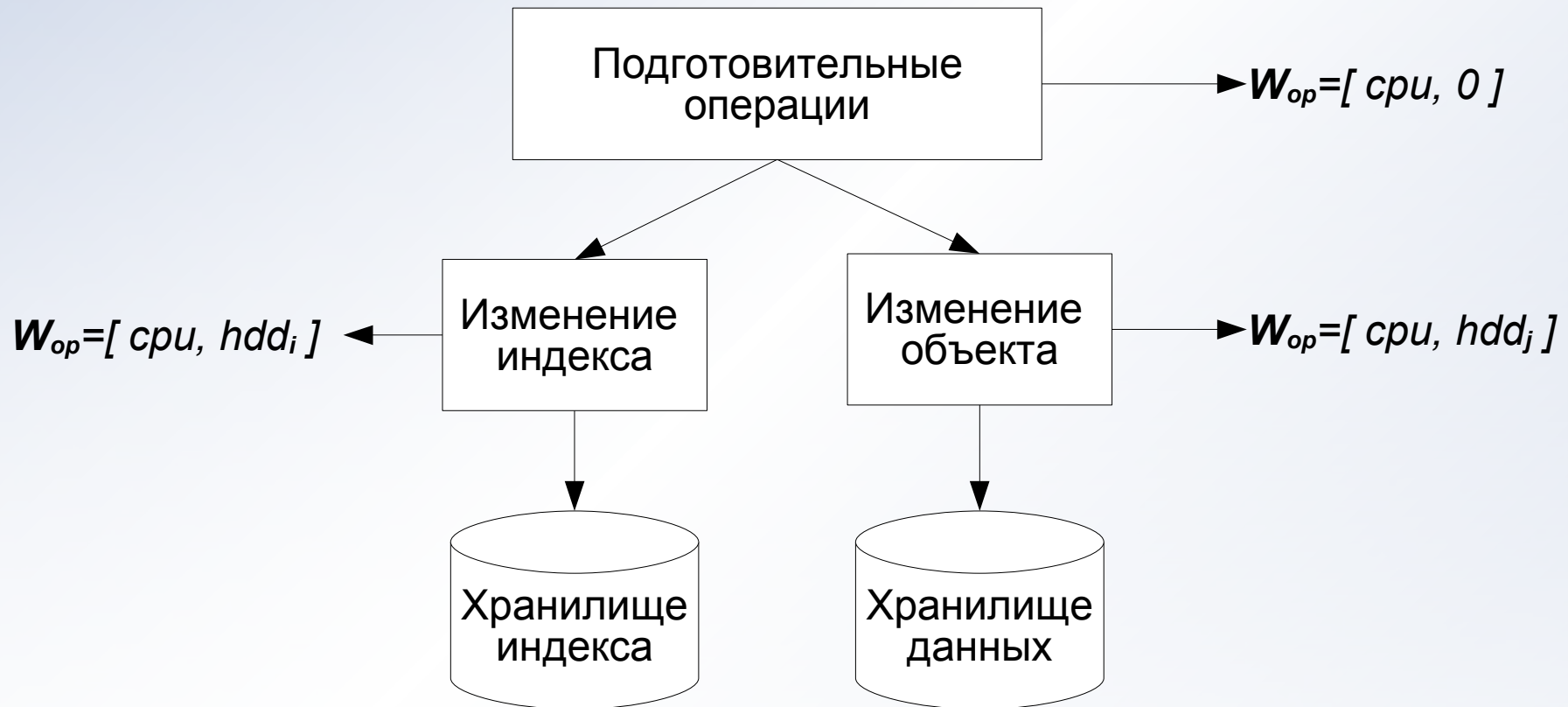
²Индексировать все реквизиты нового объекта

²Вычислить различия индексов

²Модифицировать хранилище индексов

Модификация последовательности выполнения

Параллельная реализация *IntChange*



Выявление фрагментов кода, подлежащих распараллеливанию

Пример: операция Search:

Получить список индексов, по которым производится поиск

Для всех индексов с типом «Текст»

Получить список индексов – потомков запрошенного типа

Цикл для всех индексов – потомков

Выполнить подзапрос по одному индексу

Объединить результат

Конец цикла всех индексов – потомков

Выполнить ранжирование результата

Конец цикла для индексов с типом «Текст»

Цикл для всех индексов

Получить список индексов – потомков запрошенного типа

Цикл для всех индексов – потомков

Выполнить подзапрос по одному индексу

Объединить результат

Конец цикла всех индексов – потомков

Выполнить логическую операцию с предыдущими результатами

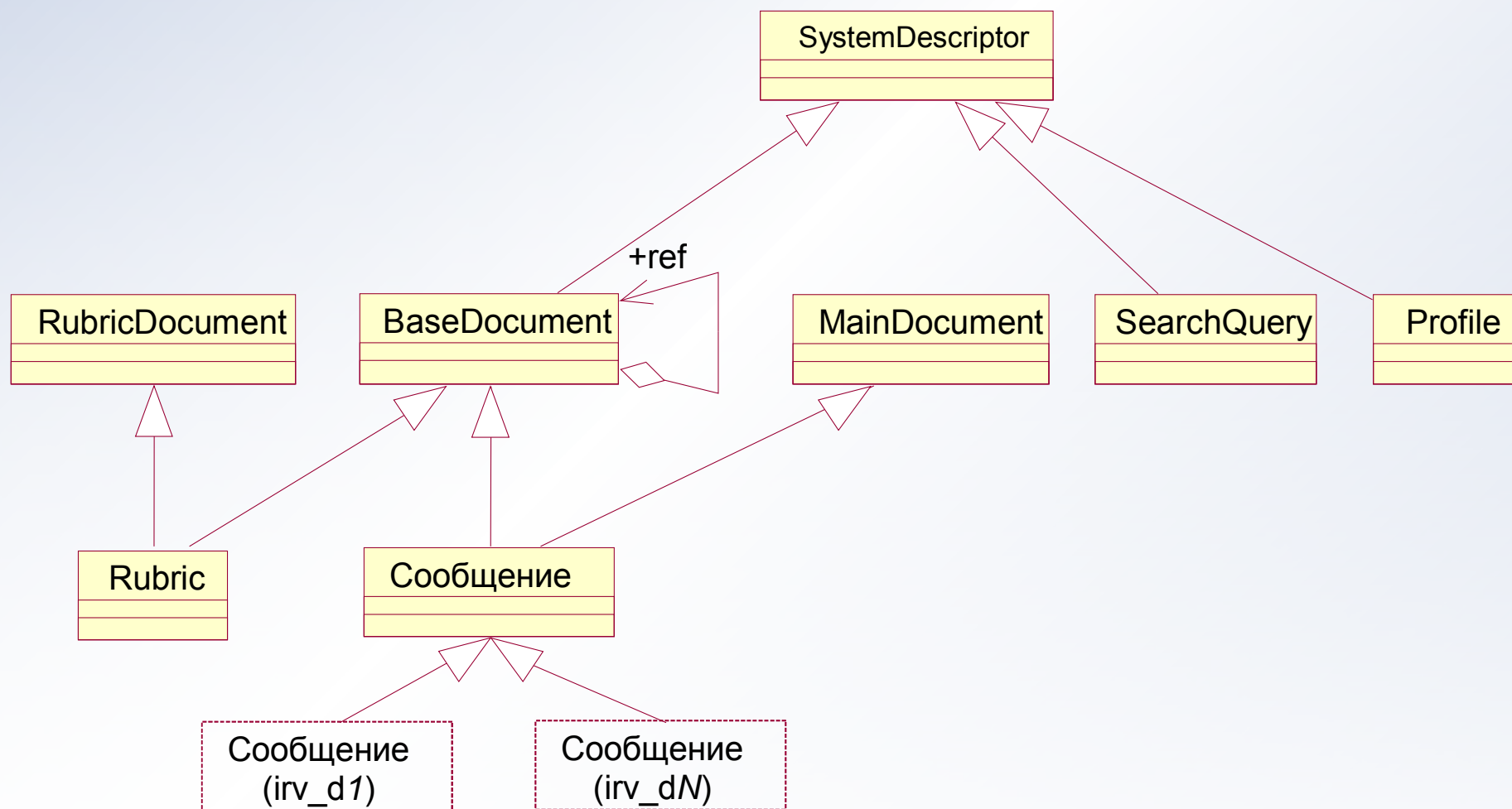
Конец цикла всех индексов

Сортировать результат поиска

Читать значения указанных реквизитов для списка найденных объектов

Схема данных ИПС «Обзор СМИ» на основе ОСУБД ODB-Jupiter

Выборка данных с учетом объектного наследования



Модификация последовательности выполнения

Параллельная реализация *Search*



Определение состава векторов состояний ВС и веса операций

Конфигурация
ВС

Конфигурация
СУБД

↓
Определение размерности
векторов

→ $W_{sys} = [cpu, \{hdd\}, \{mem\}]$

↓
Определение смысла координат
векторов (абсолютные значения или
факт использования)

$W_{sys} = [N, \{p_i\}, \{V\}]$

↓
Определение априорного веса
операций и начального значения
вектора доступных ресурсов

W_{op}

$cpu - 1$, если используется

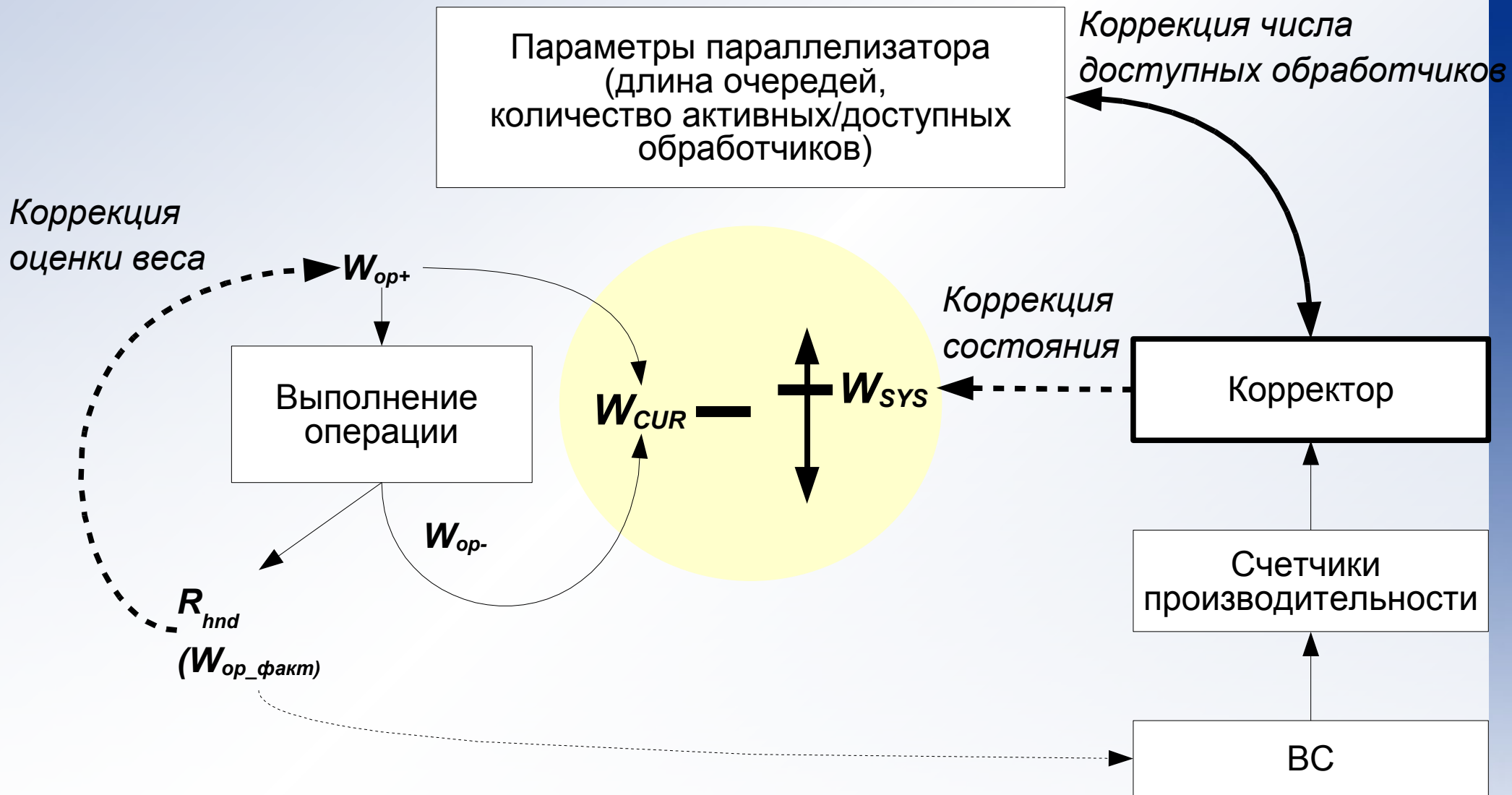
$hdd_i - p_i/m$, если чтение

p_i , если запись

$mem -$ оценка объема ОЗУ для
выполнения

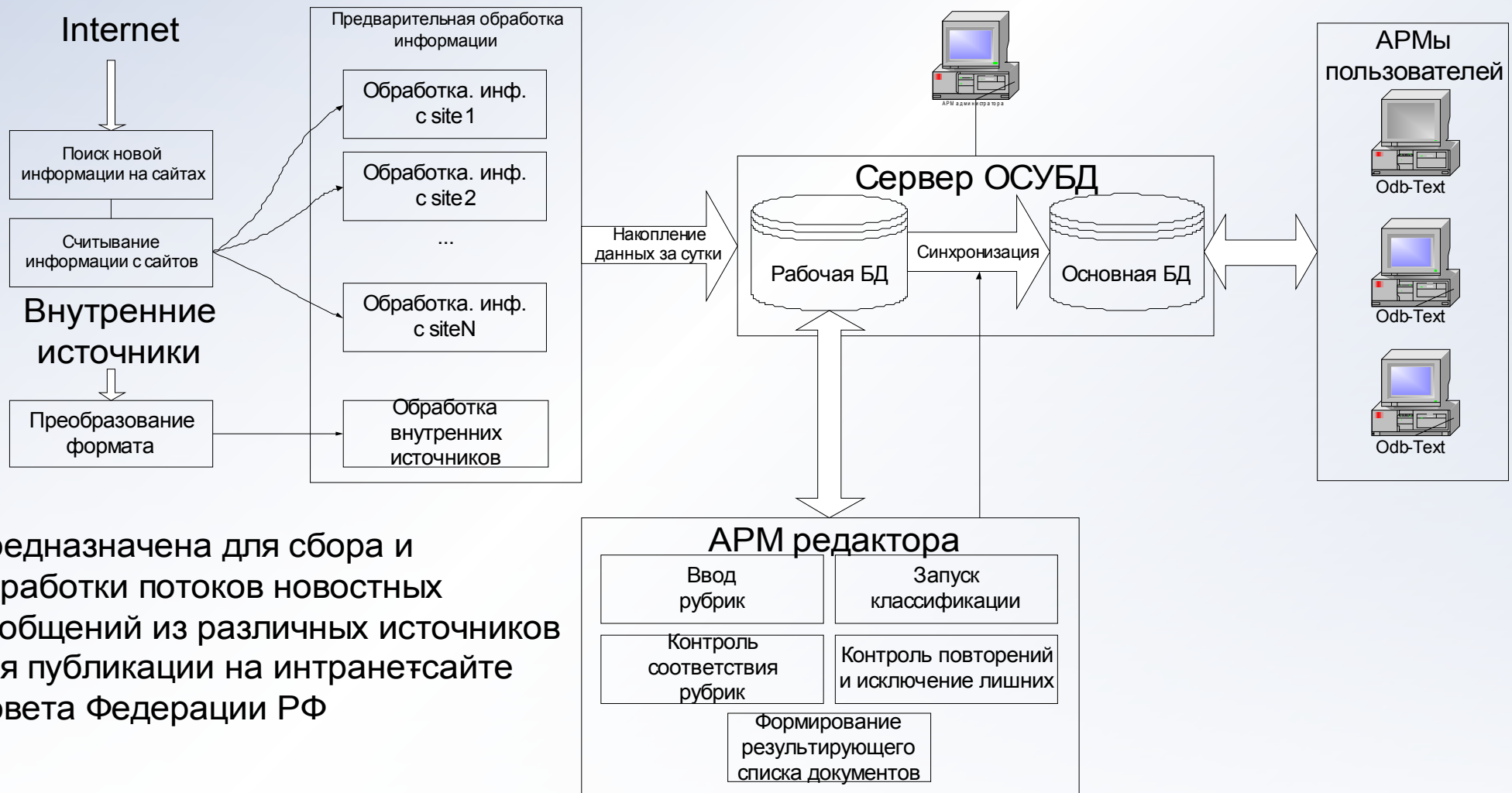
...

Определение алгоритма коррекции и оценки состояния ВС



ИПС на основе ОСУБД ODB-Jupiter

Структура ИПС обработки потоков новостных сообщений «Обзор СМИ»



Предназначена для сбора и обработки потоков новостных сообщений из различных источников для публикации на интранетсайте Совета Федерации РФ

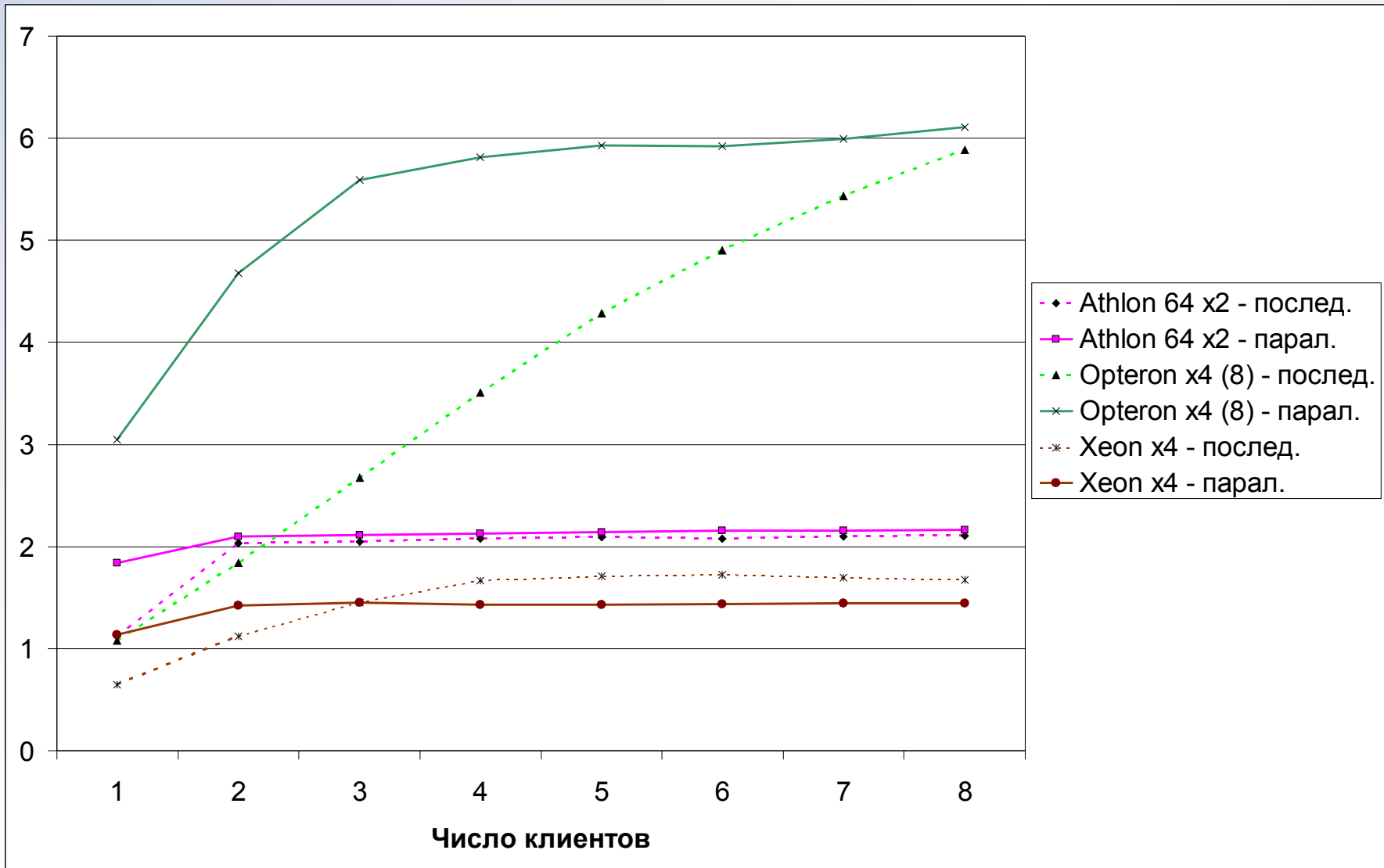
Результаты работы улучшенной ОСУБД

База документов: 8 хранилищ по 30 тыс. сообщений.
Общий объем - около 240 тыс., размер базы – 4 ГБ
(Выполнение полнотекстового поиска)

Конфигурация ВС	Процент снижения времени отклика от числа клиентов			
	1	2	3	4
Процессор AMD Athlon 64 X2 4400+ ОЗУ 2 ГБ, размер базы – 4 ГБ	37	-2	-2	-2
Процессор 4 x Opteron 880 2.4 ГГц ОЗУ 16 ГБ, размер базы – 4 ГБ	65	51	41	31
Процессор 4 x Intel Xeon 2,6 ГГц ОЗУ 1024 МБ, размер базы – 4 ГБ	43	25	-8	-15

Результаты работы улучшенной ОСУБД

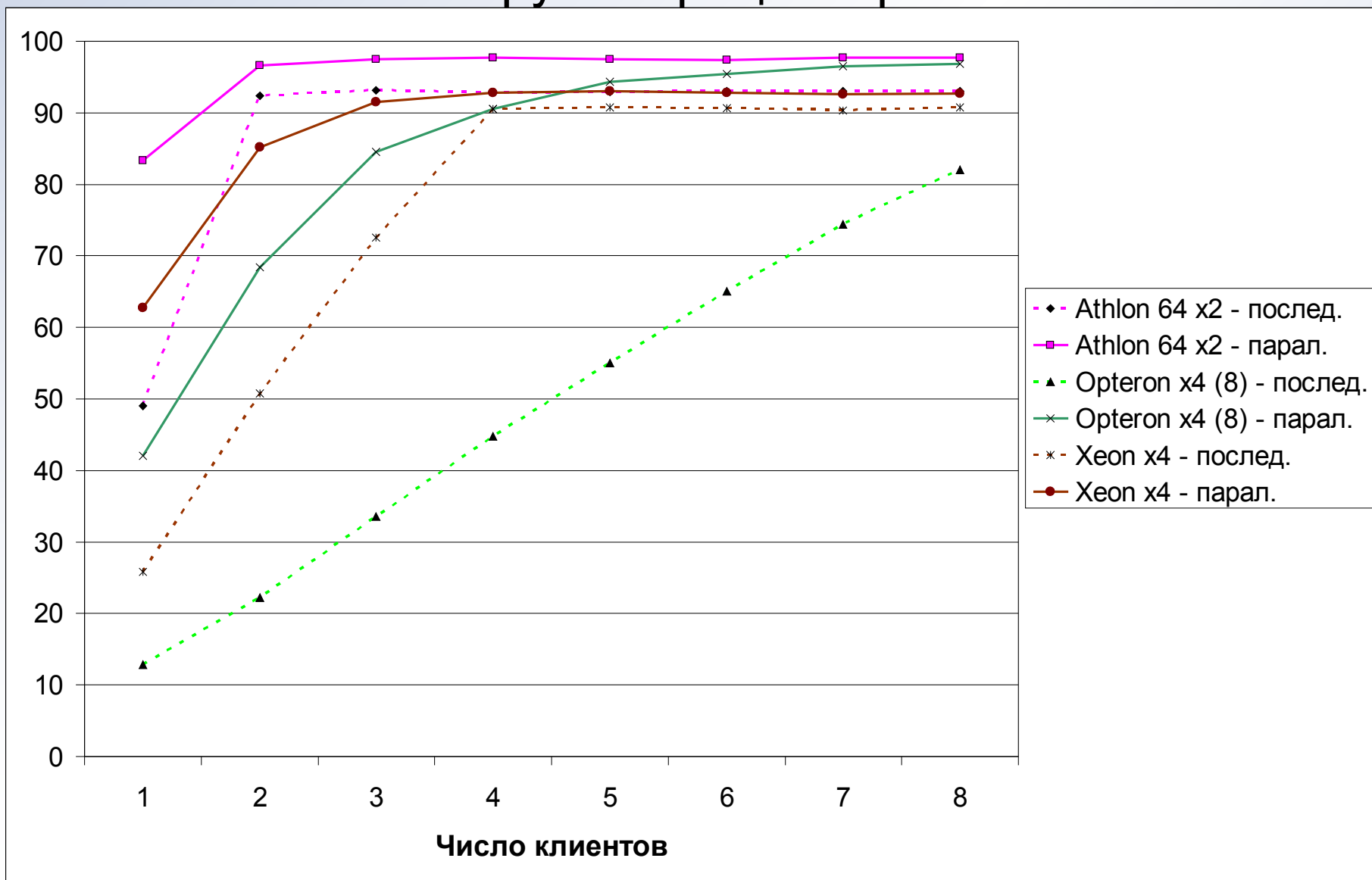
Производительность в режиме полнотекстового поиска



Результаты работы улучшенной ОСУБД

(полнотекстовый поиск)

% загрузки процессоров



Порядок преобразования операций для параллельной обработки

- Выявление фрагментов кода, подлежащих распараллеливанию;
- Определение состава ресурсов ВС, используемых совместно;
- Оценка длительности выполнения фрагментов кода в рамках выделенных фрагментов и оценка потребления ресурсов ВС при их выполнении
- Выбор оптимального разбиения на параллельные участки отобранных фрагментов кода (например при помощи алгебраического моделирования)
- Реорганизация кода программы для осуществления параллельного выполнения

Заключение

- Современное ПО должно использовать возможности параллельного выполнения процессорами
- Управляемый параллелизм с контролем по нескольким параметрам позволяет избежать деградации производительности ВС в целом, однако набор этих параметров зависит от решаемых задач
- Для эффективной работы параллелизатора необходимо прогнозирование состояния ВС в реальном времени, что в общем виде не реализовано
- Отсутствуют единые подходы определения состояния ВС для разных операционных систем

Спасибо за внимание

Самарев Роман Станиславович
samarev@acm.org