

# Параллельная симуляция. Часть 3

## Курс «Программное моделирование вычислительных систем»

Григорий Речистов  
[grigory.rechistov@phystech.edu](mailto:grigory.rechistov@phystech.edu)

3 марта 2014 г.

- 1 Обзор
- 2 Атомарные инструкции
- 3 Модели памяти
- 4 Примеры симуляторов
- 5 Литература
- 6 Конец

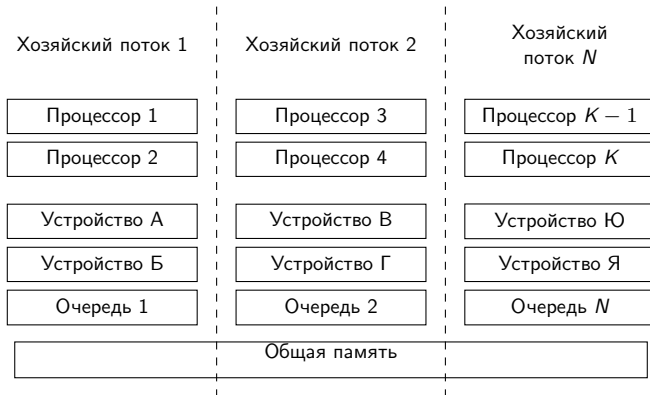
# На прошлой лекции

- Оптимистичные схемы
- Откат состояния
- Time Warp и virtual time

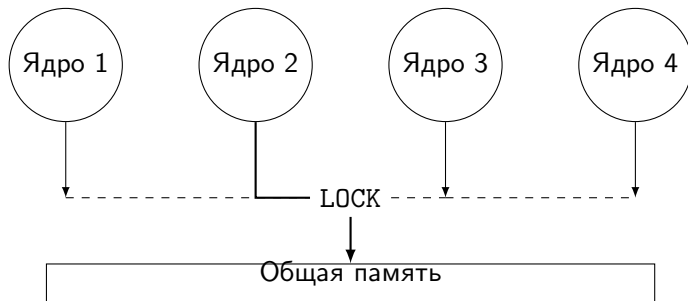
# На прошлой лекции

- Оптимистичные схемы
- Откат состояния
- Time Warp и virtual time
- Вопрос: как выполнять вывод (`printf`) в оптимистичной симуляции?

# Общая схема моделируемой системы

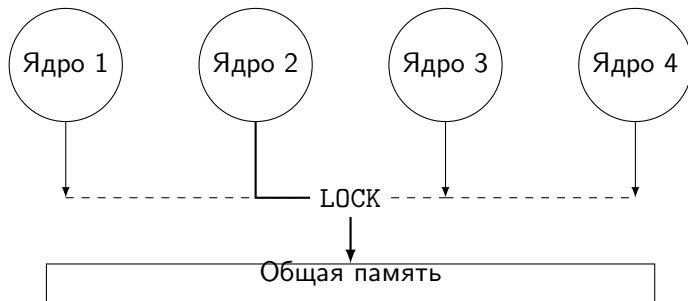


# Атомарные инструкции



- Read–Modify–Write для ячейки в памяти
- Средства реализации семафоров
- «Дорогие» для исполнения

# Атомарные инструкции



- Read-Modify-Write для ячейки в памяти
- Средства реализации семафоров
- «Дорогие» для исполнения
- Вопрос: нужны ли атомарные инструкции для однопроцессорных систем?

# Симуляция инструкций

- 1 Использование хозяйских инструкций
- 2 Использование критических секций
- 3 Использование транзакций



# Использование хозяйских инструкций

- Разные ISA для атомарных инструкций (напр. IA-32 — больше 10 инструкций, ARM — две)
- Не все атомарные операции одинаково *сильны* (consensus number) [1]
  - $\infty$  — Mem-Mem, CAS, LL/SC
  - 2 — TAS, SWAP, FAA
  - 1 — атомарное чтение и запись
- Метод наиболее удобен в случаях совпадения архитектур хозяина и гостя

# Использование критических секций

- Хозяйская критическая секция для моделирования одной атомарной операции

# Использование критических секций

- Хозяйская критическая секция для моделирования одной атомарной операции ... но это не работает
- Пример [2]: взятие семафора с помощью CAS, освобождение — с помощью атомарной записи

Процессор 1

sem\_unlock:

`*addr = 0`

Гонка  
данных

`T0 = r10  
T1 = *addr  
*addr = T0  
r10 = T1`

Процессор 2

sem\_lock:

```
try:  
  r10 = 1  
  xchg addr, r10  
  if (r10 == 0 )  
    goto success  
fail:  
  pause  
  if (*addr != 0)  
    goto fail  
  goto try  
success:
```

# Использование транзакций

- Не предотвращать, а детектировать гонки данных
- Повторять попытку атомарной операции в случае неуспеха
- Используются хозяйские инструкции CAS или (лучше) LL/SC

# Консистентность памяти

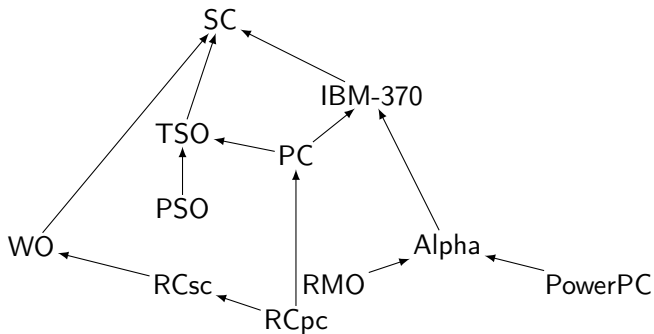
- Между процессорами и ОЗУ может лежать несколько буферов (кэши, очереди)
- Возможны ситуации, в которых разные процессоры «видят» разные значения для одних и тех ячеек
- Правила, которые определяют допустимые порядки видимости значений для архитектуры — модель консистентности памяти

# Консистентность памяти

- Между процессорами и ОЗУ может лежать несколько буферов (кэши, очереди)
- Возможны ситуации, в которых разные процессоры «видят» разные значения для одних и тех ячеек
- Правила, которые определяют допустимые порядки видимости значений для архитектуры — модель консистентности памяти
- Модели консистентности различаются между собой

# Отношение строгости

Из [3]



# Что же делать? Барьеры памяти

- Устанавливают частичный порядок для операций
- Т.е. какие из доступов в каком направлении могут опережать соседние
- Чтение, запись, доступ к устройствам, чтение инструкций
- Примеры инструкций: `sfence`, `lfence`, `mfence`; `mf`, `ld.acq`, `ld.rel`; `eioeio`, `sync`; `cuid`
- Атомарные инструкции не обязательно являются барьерами!



# Сравнение архитектур

Из [4]

|                      | Loads Reordered After Loads? | Loads Reordered After Stores? | Stores Reordered After Stores? | Stores Reordered After Loads? | Atomic Instructions Reordered With Loads? | Atomic Instructions Reordered With Stores? | Dependent Loads Reordered? | Incoherent Instruction Cache/Pipeline? |
|----------------------|------------------------------|-------------------------------|--------------------------------|-------------------------------|---|--|----------------------------|--|
| Alpha                | Y                            | Y                             | Y                              | Y                             | Y   | Y  | Y                          | Y                                      |
| AMD64                |                              |                               |                                | Y                             |   |  |                            |  |
| ARMv7-A/R            | Y                            | Y                             | Y                              | Y                             | Y   | Y  | y                          | Y                                      |
| IA64                 | Y                            | Y                             | Y                              | Y                             | Y   | Y  |                            | Y                                      |
| (PA-RISC)            | Y                            | Y                             | Y                              | Y                             |   |  |                            |  |
| PA-RISC CPUs         |                              |                               |                                |                               |   |  |                            |  |
| POWER <sup>TM</sup>  | Y                            | Y                             | Y                              | Y                             | Y   | Y  |                            | Y                                      |
| (SPARC RMO)          | Y                            | Y                             | Y                              | Y                             | Y   | Y  |                            | Y                                      |
| (SPARC PSO)          |                              |                               | Y                              | Y                             |   | Y  |                            | Y                                      |
| SPARC TSO            |                              |                               |                                | Y                             |   |  |                            | Y                                      |
| x86                  |                              |                               |                                | Y                             |   |  |                            | Y                                      |
| (x86 OOSTore)        | Y                            | Y                             | Y                              | Y                             |   |  |                            | Y                                      |
| zSeries <sup>®</sup> |                              |                               |                                | Y                             |   |  |                            | Y                                      |

# Практические параллельные симуляторы

- Simics
- Graphite
- SimOS
- Coremu
- Pqemu
- BigSim
- DynamoRIO

# Дополнительные вопросы параллельной симуляции

- Параллельная двоичная трансляция
- Распределённая общая память

# Дополнительные вопросы параллельной симуляции

- Параллельная двоичная трансляция
- Распределённая общая память
- Почему параллельная симуляция настолько сложна?

# Дополнительные вопросы параллельной симуляции

- Параллельная двоичная трансляция
- Распределённая общая память
- Почему параллельная симуляция настолько сложна?  
*«...необходимо определить, может или нет сообщение  $E_1$  быть обработано одновременно с  $E_2$ . Но каким образом узнать, влияет или нет  $E_1$  на  $E_2$ , без его симуляции?»*

# Рекомендуемая литература I



*Maurice Herlihy*. "Wait-Free Synchronization" <http://cs.brown.edu/~mph/Herlihy91/p124-herlihy.pdf>



*Zhaoguo Wang et al.* COREMU: a Scalable and Portable Parallel Full-System Emulator [http://ppi.fudan.edu.cn/\\_media/publications%3Bcoremu-ppopp11.pdf](http://ppi.fudan.edu.cn/_media/publications%3Bcoremu-ppopp11.pdf),



*Kourosh Gharachorloo* Memory Consistency Models for Shared-Memory Multiprocessors.  
<http://infolab.stanford.edu/pub/cstr/reports/csl/tr/95/685/CSL-TR-95-685.pdf>



*Paul E. McKenney* Memory Barriers: a Hardware View for Software Hackers <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.152.5245>

## Рекомендуемая литература II



*Jiun-Hung Ding et al.* PQEMU: A Parallel System Emulator Based on QEMU

<http://dx.doi.org/10.1109/ICPADS.2011.102>



*Intel Corporation* A Formal Specification of Intel® Itanium® Processor Family Memory Ordering

# Спасибо за внимание!

Материалы курса выкладываются на сайте

[http://iscalare.mipt.ru/material/course\\_materials/](http://iscalare.mipt.ru/material/course_materials/)

*Замечание:* все торговые марки и логотипы, использованные в данном материале, являются собственностью их владельцев. Представленная здесь точка зрения отражает личное мнение автора, не выступающего от лица какой-либо организации.