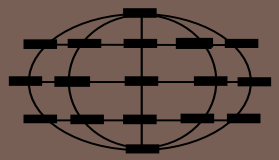
Институт Вычислительной Математики и Математической Геофизики
Лаборатория Синтеза Параллельных Программ

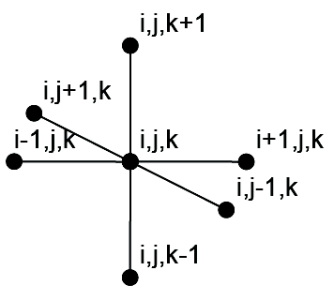


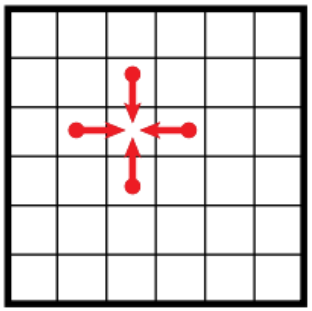
# Анализ и моделирование производительности трафаретных вычислений на мультипроцессоре

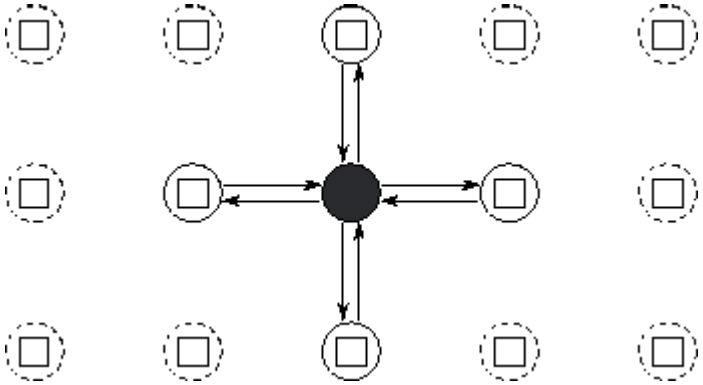
Константин Калгин
Киреев Сергей

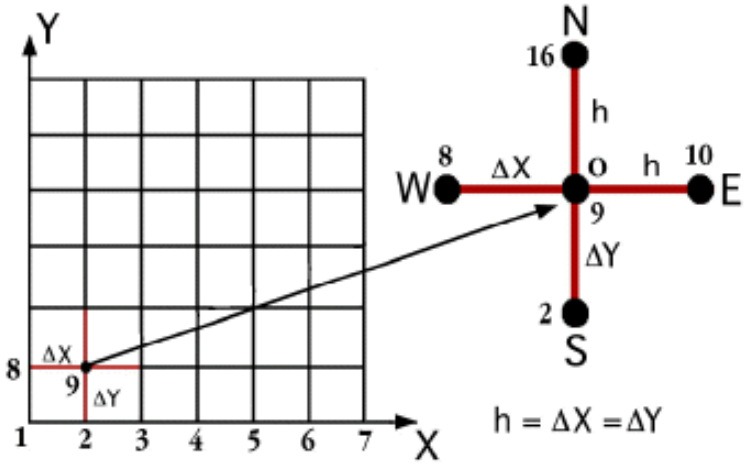
# Трафаретные вычисления
## Stencil computations

- Явная разностная схема
- Синхронный клеточный автомат



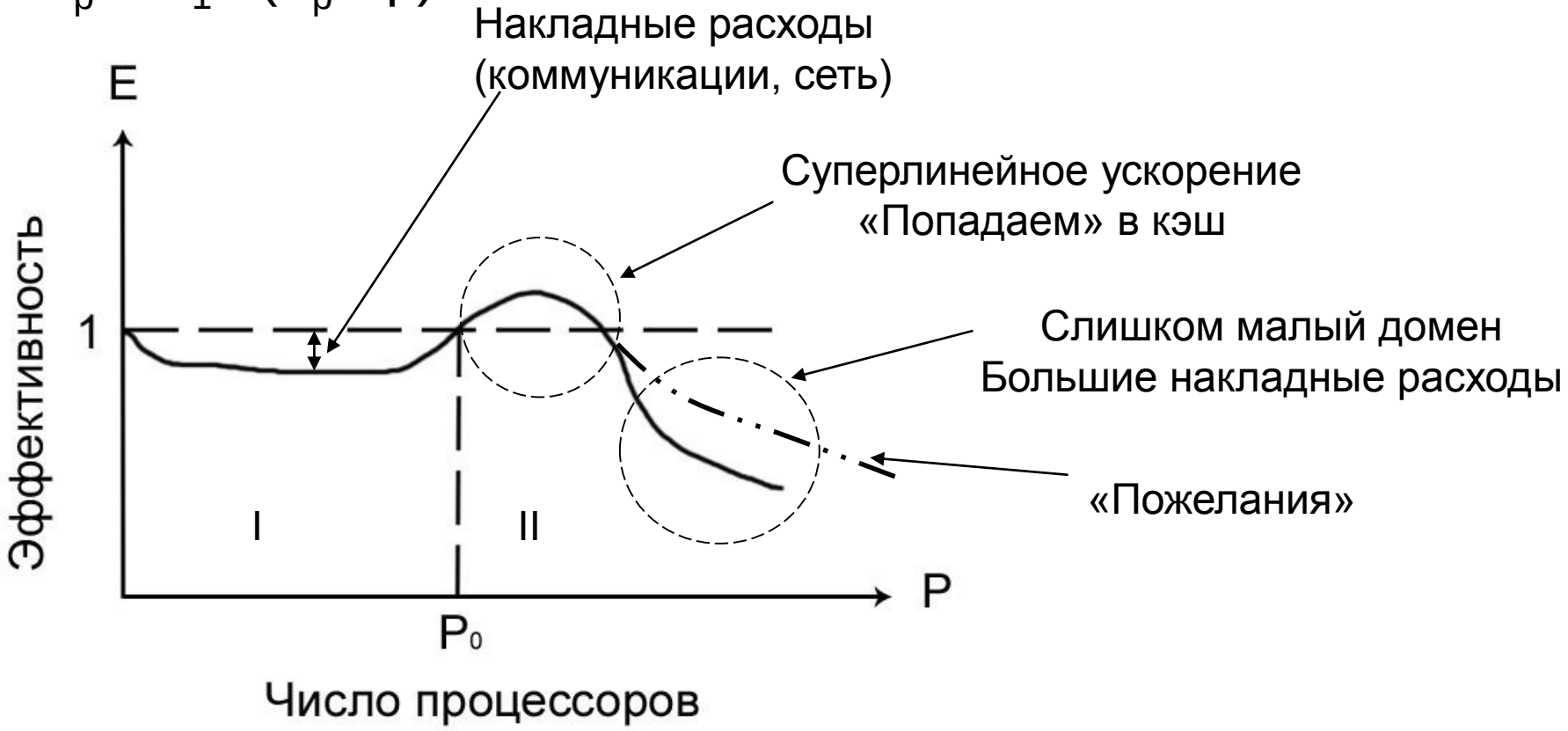







# Типичный график эффективности (размер задачи не изменяется)

$E_p = T_1 / (T_p * p)$



# Постановка задачи

- Проанализировать и построить модель производительности трафаретных вычислений для ccNUMA архитектур в случае, когда домен помещается в кэш (зона II)

$$T = T_{seq} + T_{bar} + T_{comm}$$

- Тестирование:
  - Intel Westmere (2 x 6 ядер) nks30t, SL_q
  - Intel SandyBridge (2 x 8 ядер) mvs1p1

- Вычислительные схемы 2D:
  - 5- и 9-точечные
  - невзвешенные и взвешенные средние
  - float и double

# Общие условия замера времени

- На каждом ядре запускается счёт
  - это выравнивает частоту ядер (TurboBoost)
- Перед замерами времени 0.5с процессор работает вхолостую
  - это выравнивает частоту (SpeedStep)
- Время измеряется в тактах rdtsc
  - наиболее точный таймер
  - имеет минимальные накладные расходы
- В качестве результата берётся среднее по 1000 запускам
  - отсеиваются значения на 50% превосходящие минимум
    это позволяет минимизировать влияния системных прерываний
- Компиляция: icpc -O3 -march=native -fno-inline -unroll0

# Последовательная реализация

$$T_{seq}(x,y) = A + By + Cxy$$

- A - время вызова процедуры, реализующей итерацию

$$A = A_1$$

~10ns

- B - время, затрачиваемое на переход с одной строки домена на другую (двумерная область)

$$B = B_1 + B_2(x > 64)$$

~2ns

- C - время обработки одного узла домена

$$C = C_1 + C_2(size \in L_2) + C_3(size \in L_3).$$

~0.5-2ns

# Барьер

- Замер времени:
  последовательная реализация + барьер,
  без коммуникаций/копирований
- Типы барьера:

| | ns |
|---|---|
| OpenMP (#pragma omp barrier) | 500 \| 1200 |
| Ожидания на volatile переменных | 200 \| 600 |
| Локальная синхронизация (не барьер) | 80 \| 250 |

  - +ещё много вариантов (иерархический, атомарные операции,потоковая запись) тестируется на студентах в рамках курса «Эфф.прогр.мультипроцессоров»

# Latency,Westmare, Intel

Чужой сокет

Свой сокет

Свой кэш

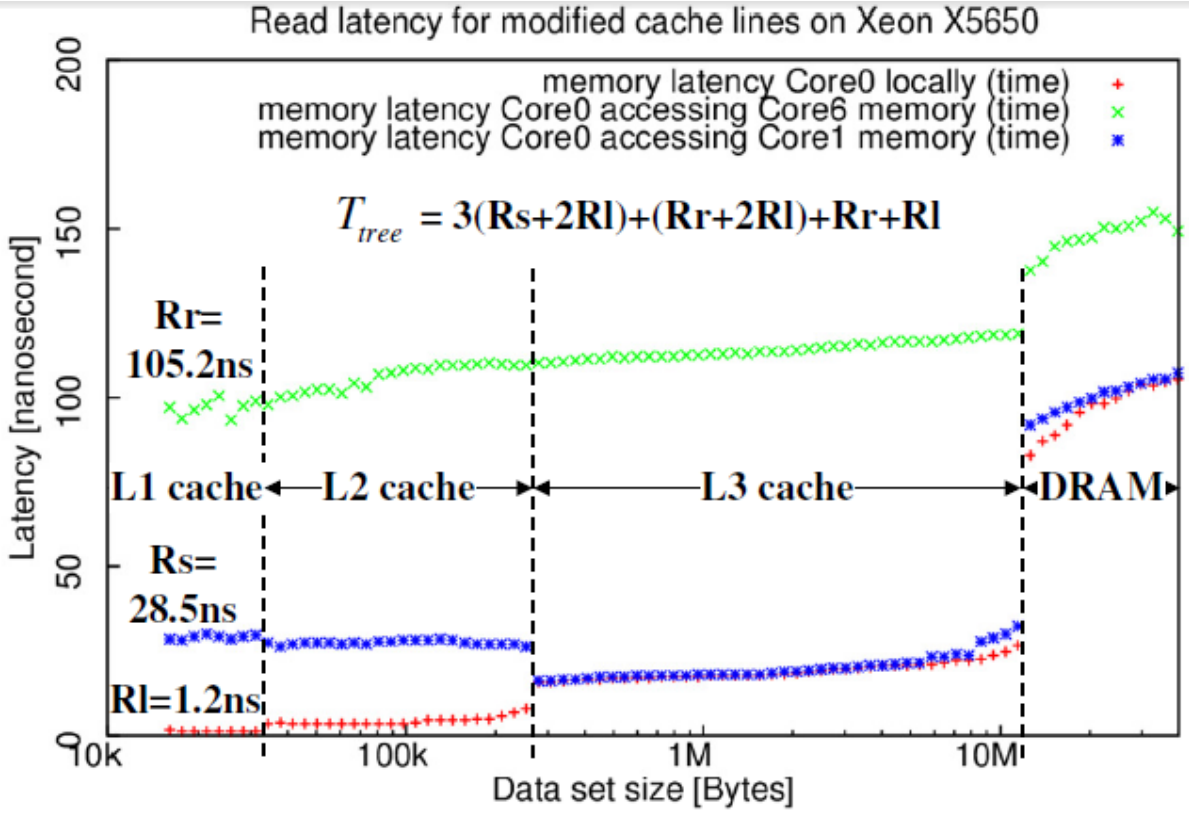
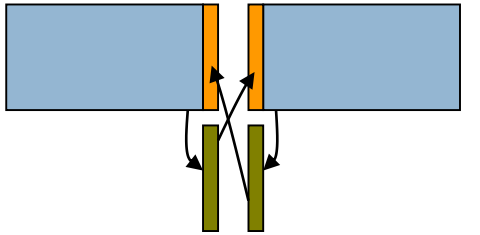


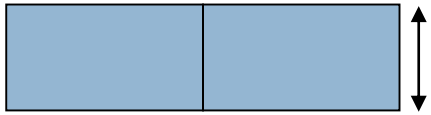

Figure 15: Modified cache line read latency on Westmere. Rl denotes latency of read local L1 cache, Rs denotes latency of read other L1 cache but within the same socket, and Rr denotes latency of read other L1 cache from remote socket.

# Границы (5nbh,a=0,float)

private




shared



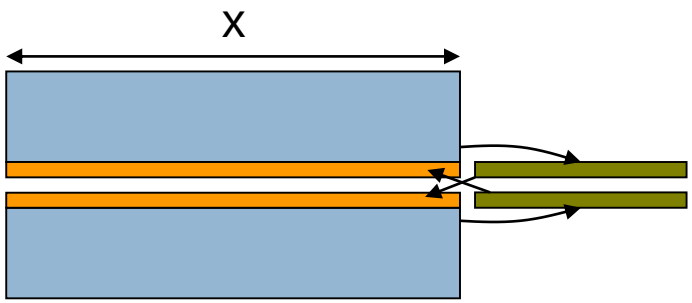

3 | 3.5 - ns

$$T_{V,P}(x,y) = Z_1 + W_1(y-1) + Qy,$$

60 | 180 - ns

$$T_{V,S}(x,y) = Z_k y,\ xy \in Lk.$$

private




0.8 | 1.2 - ns

$$T_{H,P}(x,y) = Z_1 + W_1(\mathrm{x}-1) + Q'\,\mathrm{x}$$

60 | 180 - ns

0.4 | 1.5 L1

0.22 | 0.8 L2-L3

$$T_{H,S}(x,y) = Z_k + W_k(\mathrm{x}-1),\ xy \in Lk.$$

shared

# Заключение

- Построенная модель позволяет описывать время исполнения последовательной реализации
с точностью 2-3%
- Традиционно используемый барьер встроенный в OpenMP достаточно медленный и даёт существенный вклад на малых доменах (L1), полученная реализация работет в пять (L1) и в два (L2/L3) раз быстрее
- Построена модель времени межядерного взаимодействия на границах доменов, выявлены оптимальные способы деления счетной области (shared/private)

Дальнейшие планы

- Расширить модель до MPI версии (между узлами)
- Автоматизировать получение оптимального деления на домены
- Сравнить производительность с типовыми реализациями OpenMP/MPI