System Software Research (1)

Towards Next-Generation Supercomputing

Performance Analysis and Runtimes

Performance Metrics and Tools

We've installed following performance monitoring tools into TSUBAME2.5 and can get performance data of real applications in following metrics.

Tool	Type	Metrics
Scalasca	Profiler	Time, Visit, MPI Communication, PAPI
Vampir	Tracer	Time, MPI Comm., GPU Comm., PAPI
Exana	Tracer	Memory Access (Instruction Level)
PAPI	Library	Instruction Mix, Main Memory Access

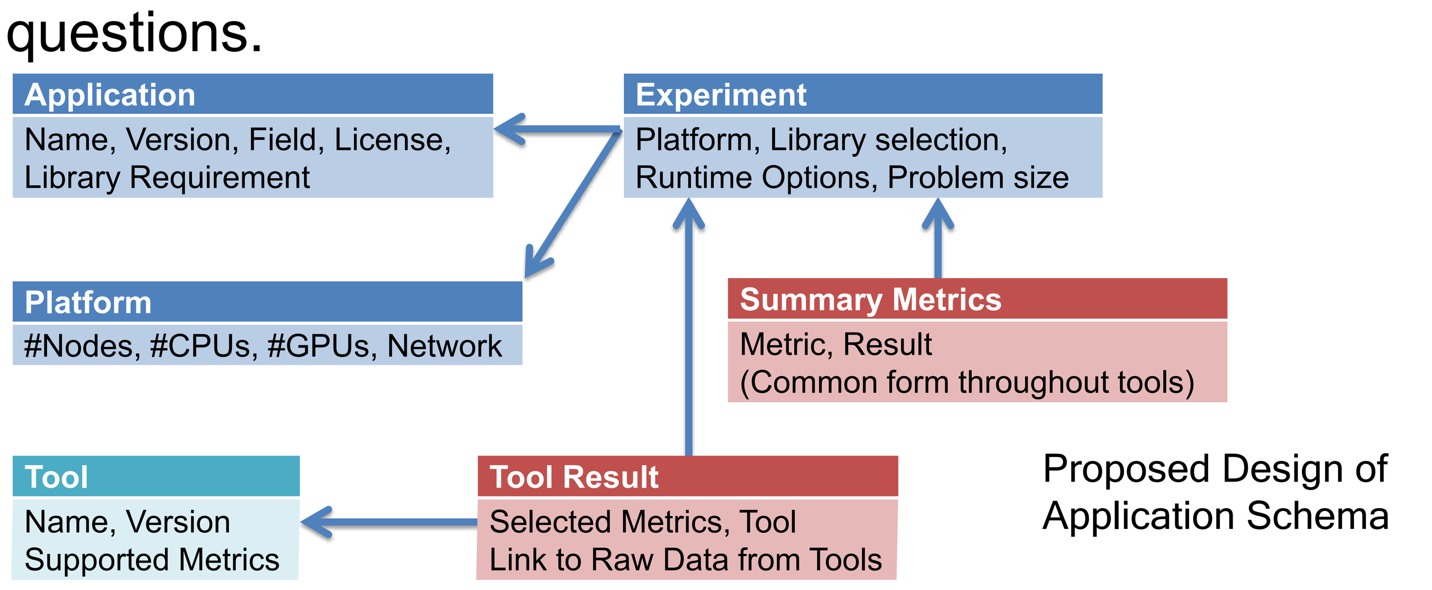
We're giving seminar for those performance tools for TSUBAME users.

Performance Repository Schema

We're building application performance repository to know following things through performance analysis.

- Which machine can solve my application well?
- Which application are suitable for our supercomputer?

We collect various performance data from many supercomputers and correlate them to answer these questions.

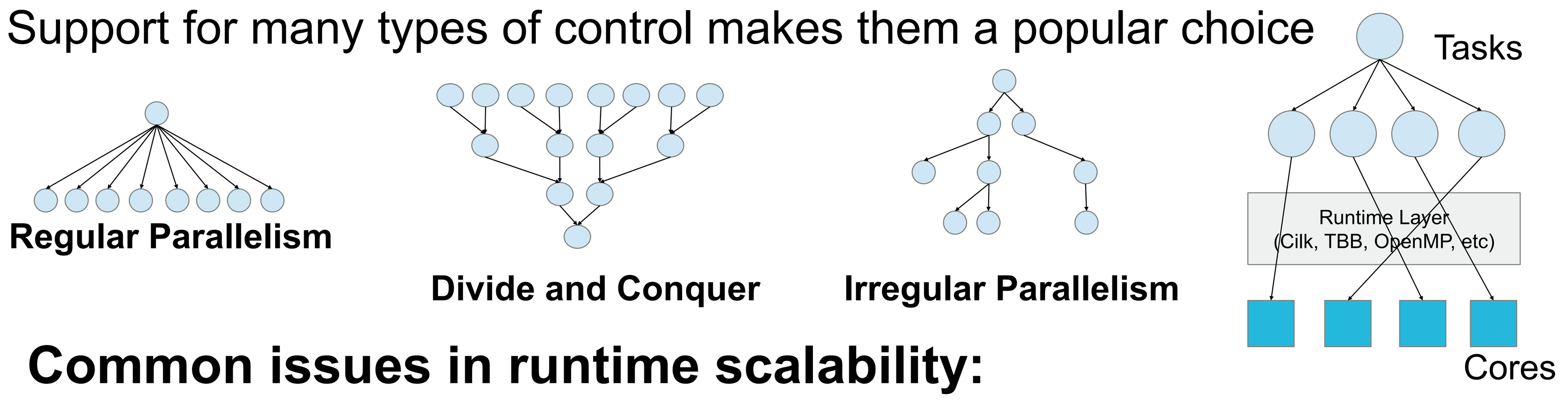


Proposed Design of Application Schema

This is collaborative work with Future Technologies Group, Oak Ridge National Laboratory.

Task Parallel Runtimes

Support for many types of control makes them a popular choice

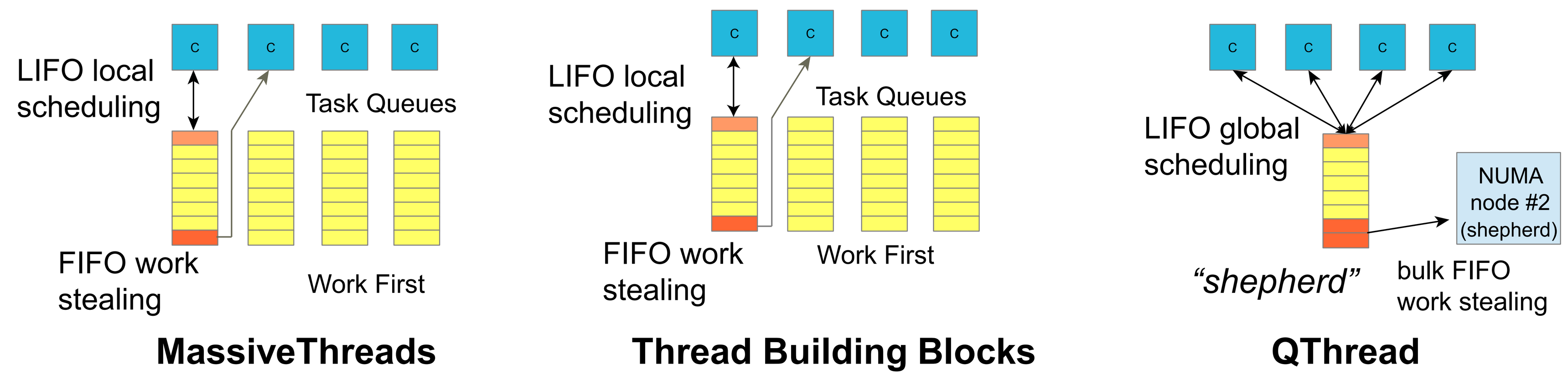


Common issues in runtime scalability:

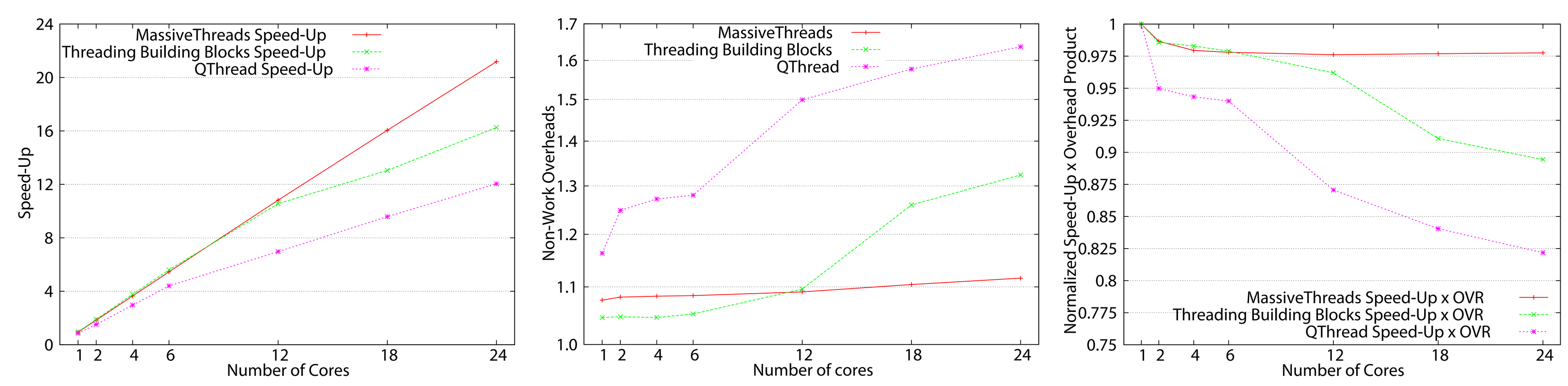
1. Runtime System overheads	2. Scheduling Constraints	3. Resource contention
Causes *parallelization overhead (PO)*	Causes *parallel idleness (PI)*	Causes *Work Time Inflation (WTI)*

$$Tserial = \sum Tkernel + \sum Tcontrol \qquad Tparallel = \frac{\sum Tkernel + \sum Tcontrol}{N} \times PI \times PO \times WTI$$

Case Study: scalability of recursive MatMul



Experiment Details: 4-socket Intel Xeon E7-4807 (1.86GHz). Matrix size 4096^2 elem (64MB)



Speed-UP — Overhead (PI x PO) — Speed-Up x OVR Product (1/WTI)

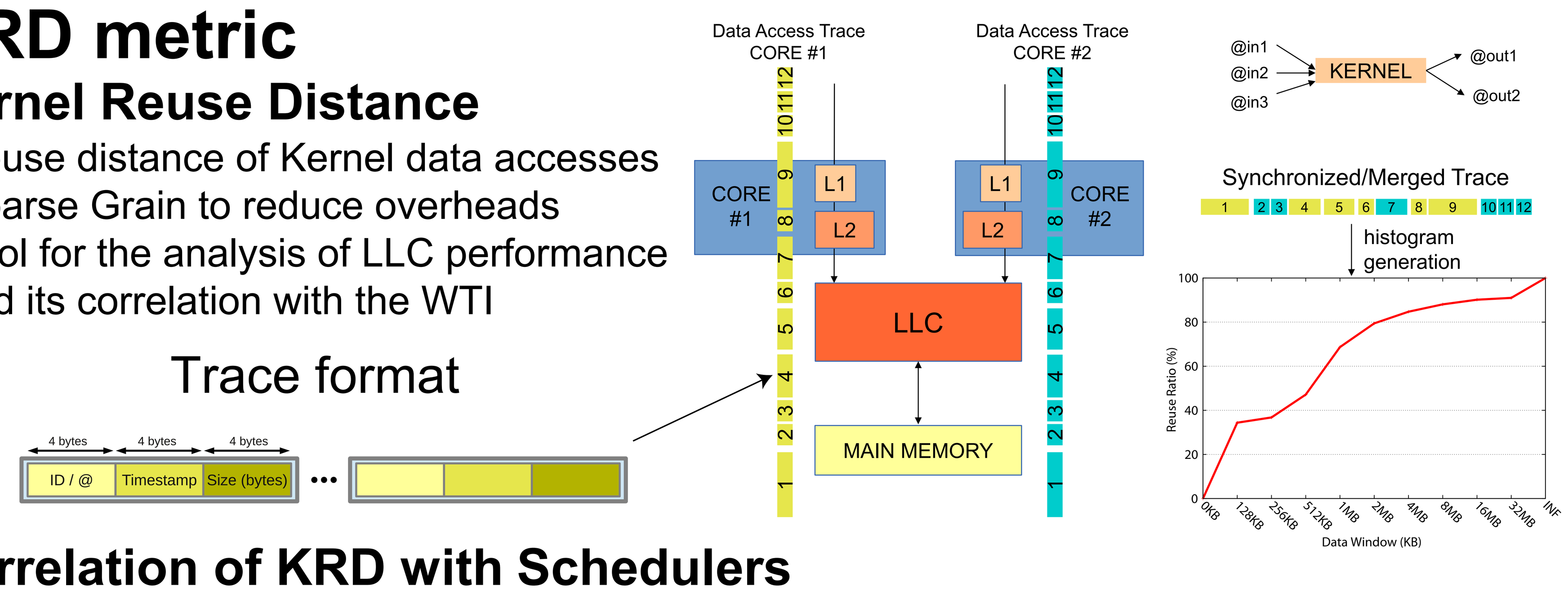
Data Reuse and Heterogeneity

KRD metric

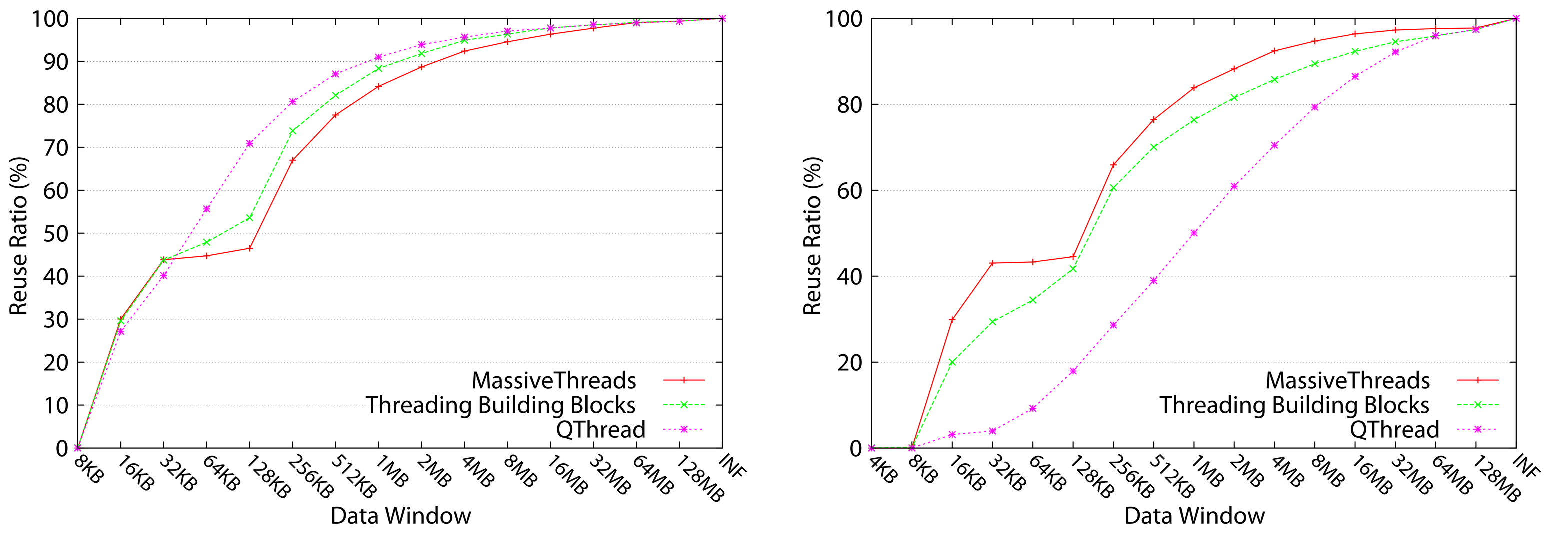
Kernel Reuse Distance

- Reuse distance of Kernel data accesses
- Coarse Grain to reduce overheads
- Tool for the analysis of LLC performance and its correlation with the WTI

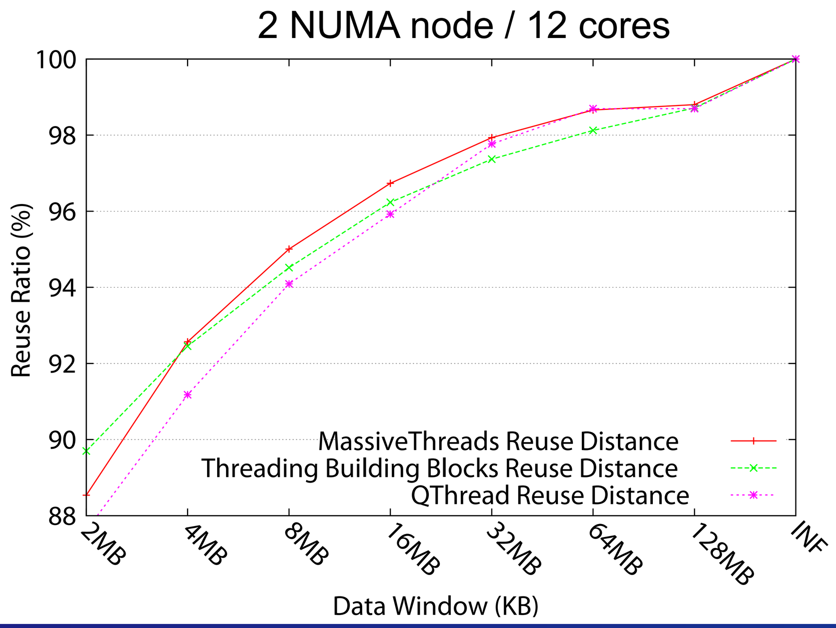
Trace format



Correlation of KRD with Schedulers



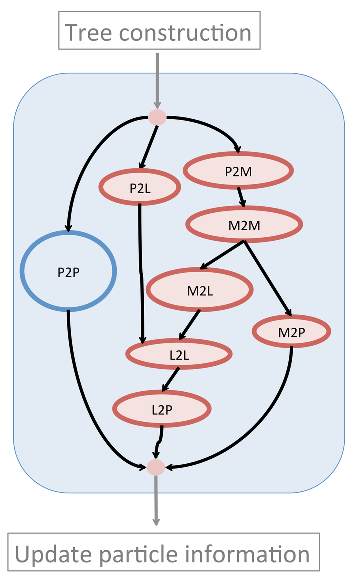
Correlation of KRD with Hardware Metrics



Runtime Library	Exec Time	LLC Misses	Kernel Time & WTI
MTH	1.653 sec	2.091×10^6	17479 ns (1.0274x)
TBB	1.721 sec	2.663×10^6	17885 ns (1.0513x)
QThread	2.564 sec	10.866×10^6	19804 ns (1.164x)

Global LIFO increases locality in single socket
MTH and TBB work stealing across node more efficient
Qthread performance on multiple sockets is hampered by runtime overheads. WTI also increases as a result

Heterogeneous Scheduling



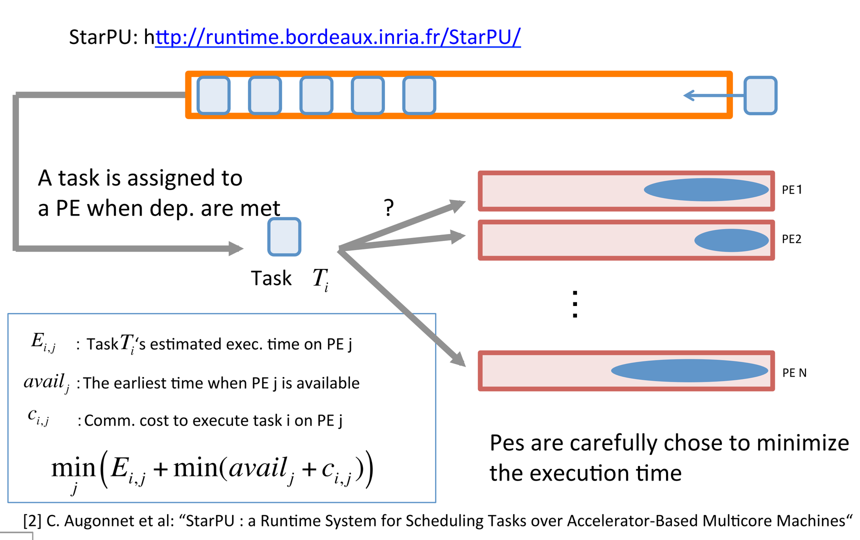
Case Study: Fast Multipole Method

1) Complex inter-task dependencies
2) Heterogeneous task runtimes
3) Input dependent control flow
4) Mix of compute/memory bound tasks

Where and when to schedule each task on a CPU/GPU system?

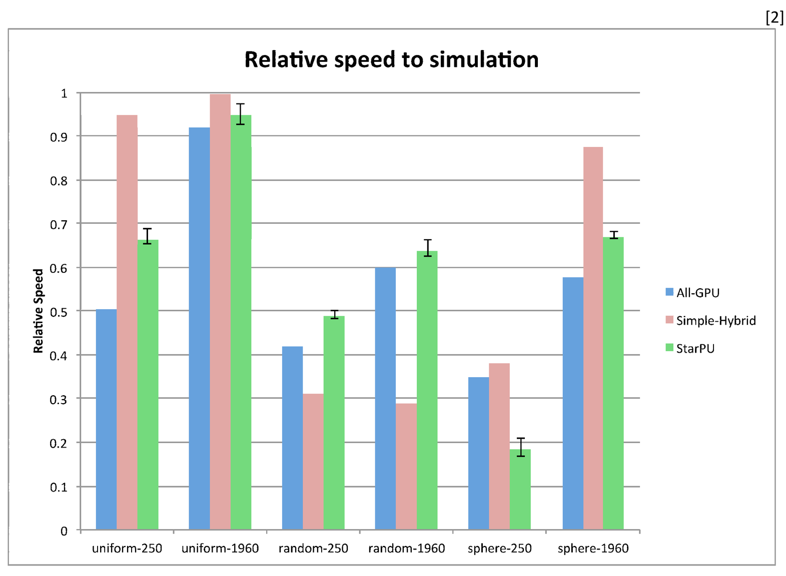
Approach:
Use dynamic task scheduler

A dynamic task scheduling engine: StarPU



Comparison of StarPU with:

1) **All-GPU**: All tasks on GPU
2) **Simple Hybrid**, P2P on GPU and remaining phases on CPU
3) **Optimal** scheduling, based on design space exploration



Lessons learned:

- Task Size Matters:
  CPUs prefer fine-grained tasks
  GPUs prefer coarse-grained tasks
- Communication / disjoint memories
- Task runtime prediciton in irregular applications is a complex problem

http://www.gsic.titech.ac.jp/sc13