Simulation Using Dynamic Schedulers

Blake Haugen

4-19-13

Introduction

- Schedulers
  - QUARK
  - StarPU
  - OmpSs/SMPSs
- Simulation Methodology
- Results
- Future Work

OmpSs/SMPSs

- Developed from StarSs
  - OmpSs
  - SMPSs
  - CellSs
  - GPUSs
- OmpSs (Latest Version)
  - Nanos++ Runtime
    - OmpSs
    - Chapel
    - OpenMP
  - Mercurium Compiler
    - Source to Source Compiler
    - C, C++, Fortran(in development)

OmpSs/SMPSs

- Entirely Based on Compiler Directives
  - Little interaction with underlying implementation
- GPU Support (Simplified Interface)
- Trace Generation
- Experimental Distributed System
- No Libraries Necessary at Runtime
- Various Scheduling Algorithms
- No Support for Multithreaded Tasks

QUARK

- Currently No GPU Support
- Distributed Tasks Supported
- Create and Insert Tasks
  - Analyze Pointers
  - Determine Dependencies
  - Schedule accordingly
- DAG Generation
- No Libraries Necessary at Runtime
- Supports Multithreaded Tasks

StarPU

- GPU Support
  - Automatic Data transfers
- Distributed Supported
  - User Defined
  - Automatic
- Multiple Interfaces
- Dependency analysis
  - Explicit Using Tags
  - Implicit

StarPU

- Data is Associated with Handles
- Various Scheduling Algorithms
- Performance Analysis
- DAG Generation
- Built-in Tracing Functions
- Libraries Necessary at Runtime
- Supports Multithreaded Tasks

Scheduler Comparison

- OmpSs/SMPSs
  - Quickest To Implement
  - Lacks Some Flexibility
  - Documentation and Examples Lacking
- QUARK
  - Small Interface
  - Some Features Still in Development
  - Several examples in PLASMA
  - Good Documentation
- StarPU
  - Enormous Feature Set
  - Good Documentation
  - Example Codes Poorly Documented

Simulation Methodology

- Create Simulated Trace
- Scheduler maintains dependencies
- No Computation is Performed
- Three Elements
  - Simulation Clock
  - Task Queue
  - Trace

Simulation Methodology

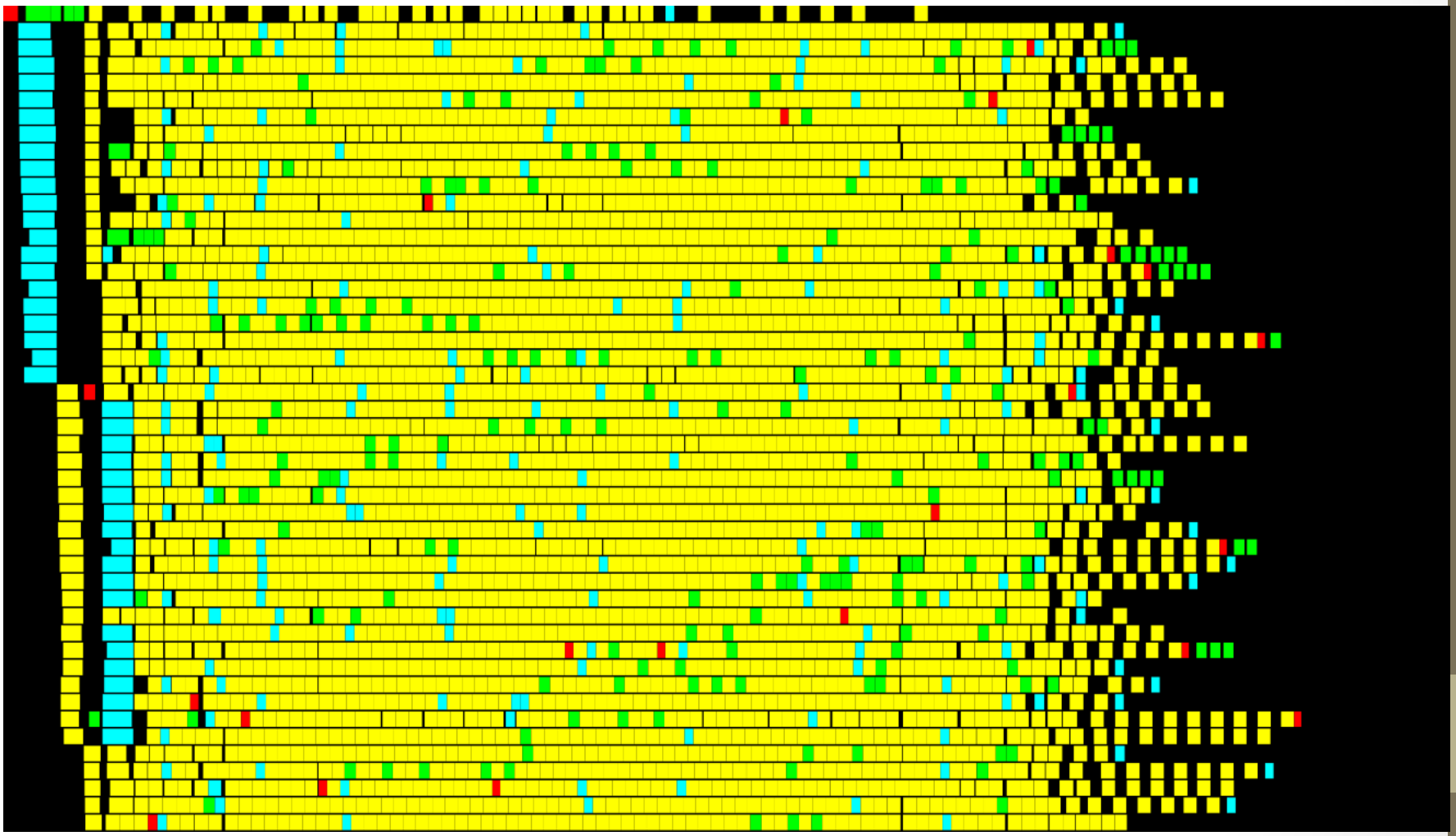
- Initialize trace and Simulation
- Replace all computation calls with simulation calls
- Maintain memory dependency inputs
- Dump Trace

Simulation Methodology

1. Calculate Task End Time
2. Insert in Task Queue
3. Wait Until Front of Queue
4. Remove from Queue
5. Update Simulation Clock
6. Insert Info in trace

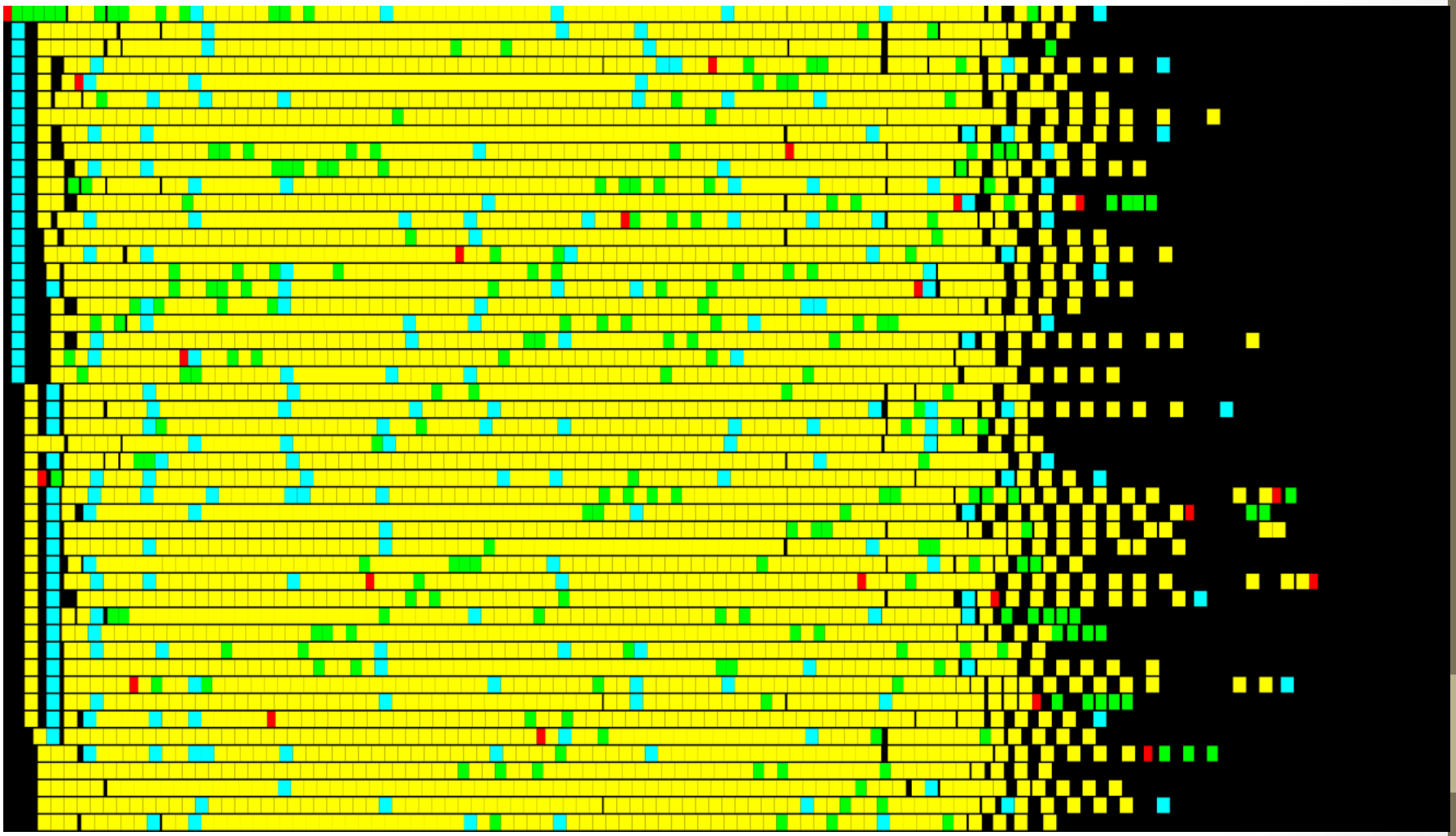
Real Trace QR

N=3096 NB=180 AMD Opteron 6180SE (48 Cores)



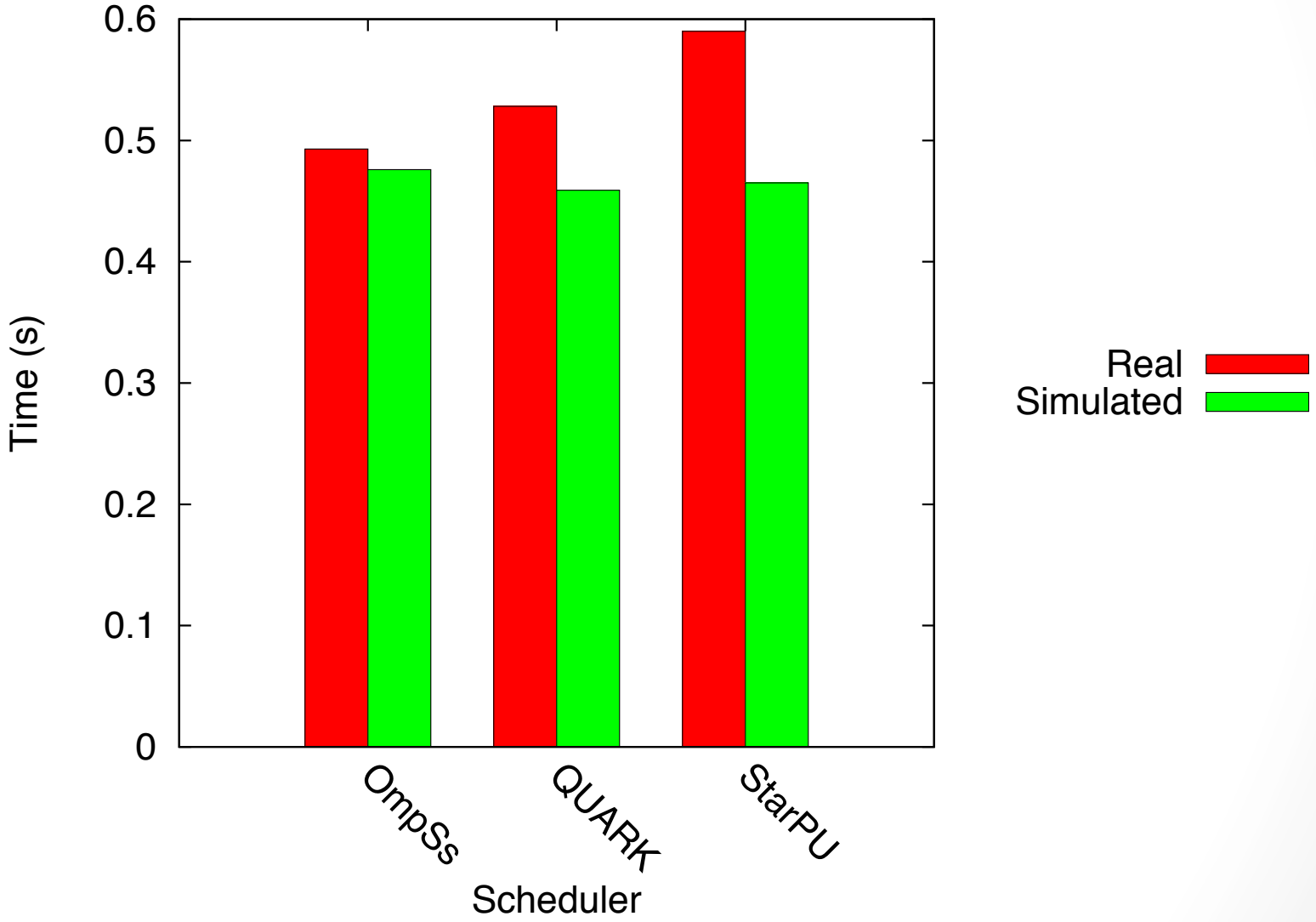
Simulated Trace QR

N=3096 NB=180 AMD Opteron 6180SE (48 Cores)



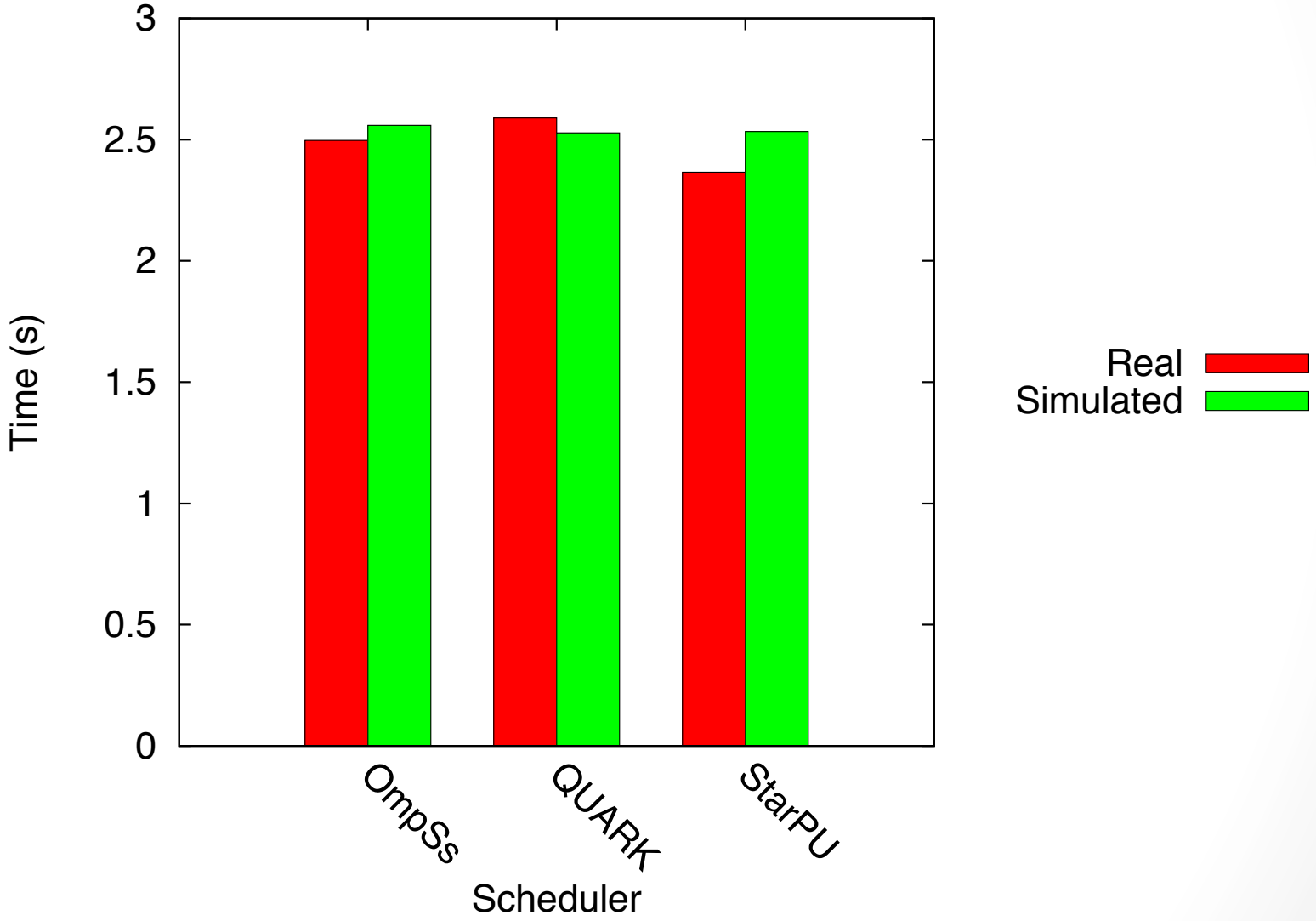
Cholesky

N=5000 NB=200 AMD Opteron 6180SE (12 Cores)



QR

N=5000 NB=200 AMD Opteron 6180SE (12 Cores)



Future Work

- Simulate LU and other workloads
- GPU Tasks
- Multithreaded Tasks
- Increase simulation speed

Conclusions

- Schedulers
- Simulation Methodology
- Results
- Future Work

Questions?