A Comparison of Library Tracking Methods in High Performance Computing

Computer System Cluster and Networking Summer Institute 2013
Poster Seminar

William Rosenberger (New Mexico Tech),
Dennis Trujillo (New Mexico State University)
Chris DeJager (Michigan Technological University)

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH UNIVERSITY

NM STATE UNIVERSITY

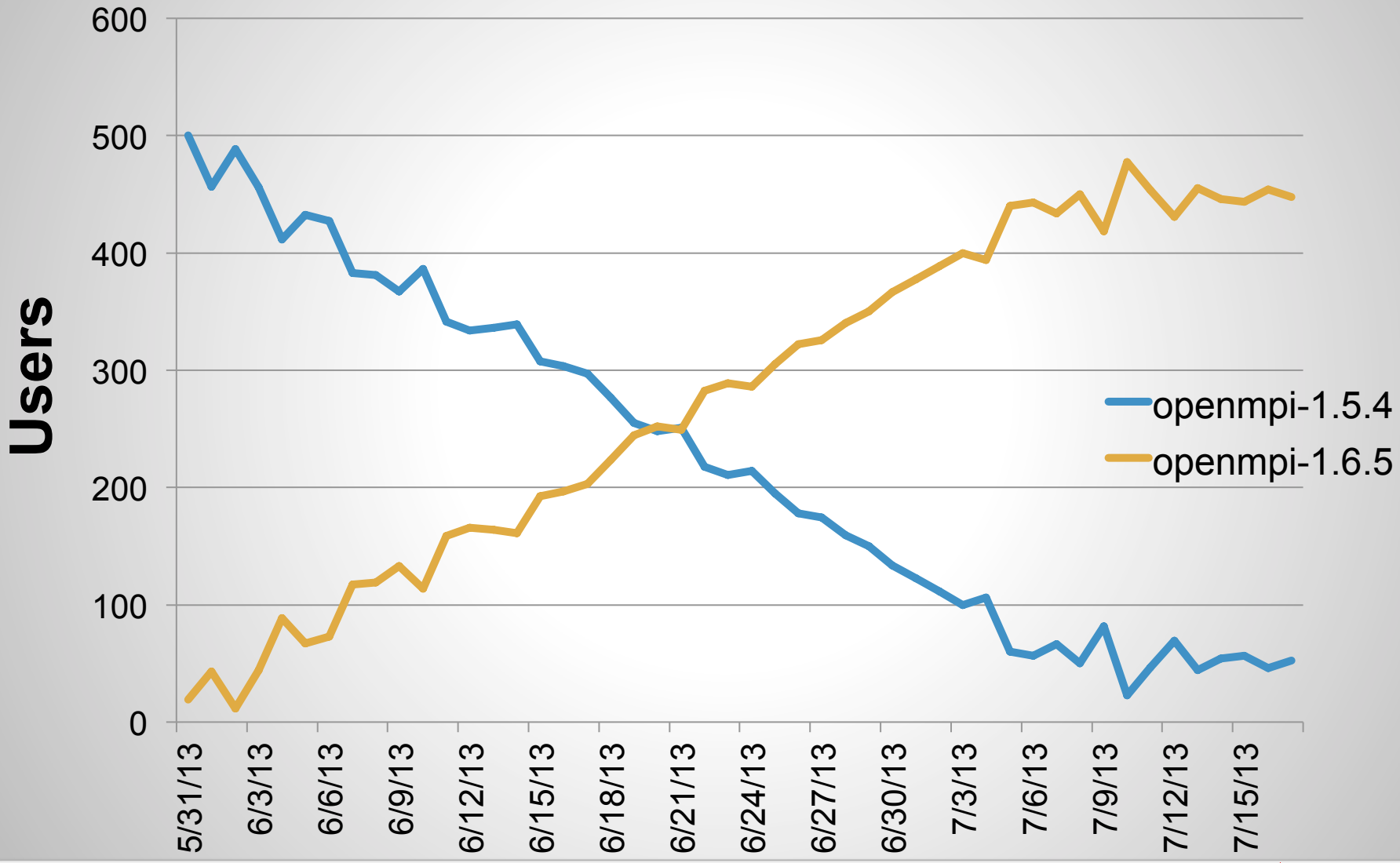
MichiganTech
Michigan Technological University

LA-UR-13-25963

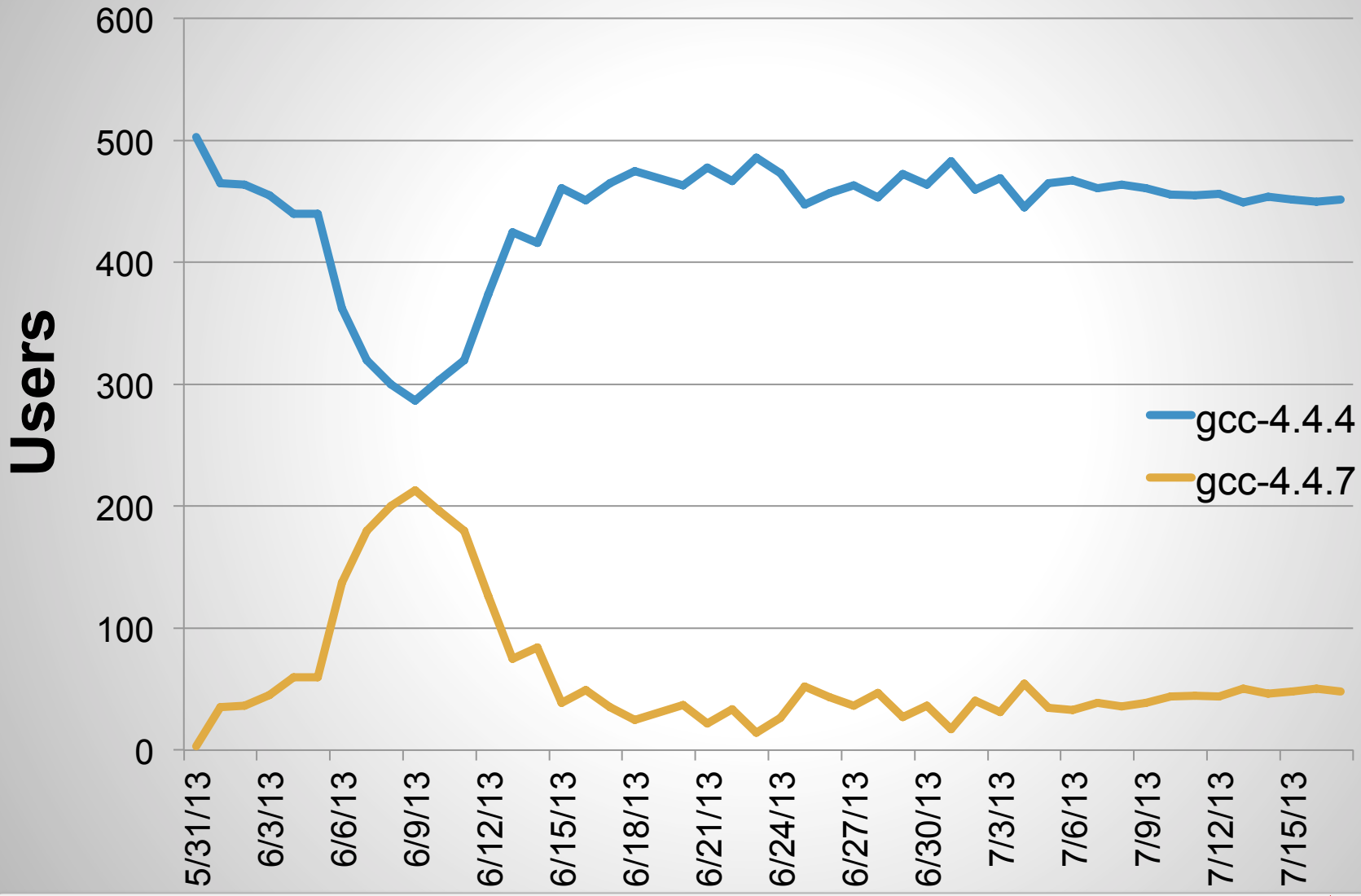
Need for a Library Tracking Utility

- HPC requires the support of many software packages, and multiple versions must be supported at the same time
  - Issues
    - License availability
    - Additional licensing cost
    - Admin support
- No existing methods of tracking user software behavior
- Current user tracking methods don't typically illustrate true use
- Eliminate less utilized software and compiler options

Simulated New Software Version Acceptance



Simulated New Software Version Rejection



Potential Tracking Solutions

Automatic Library Tracking Database

- Developed at National Institute of Computational Science for use on Cray systems
- Presented at Cray User Group 2010
- Wraps the linker 'ld' and job launcher 'aprun'
- Stores to MySQL database

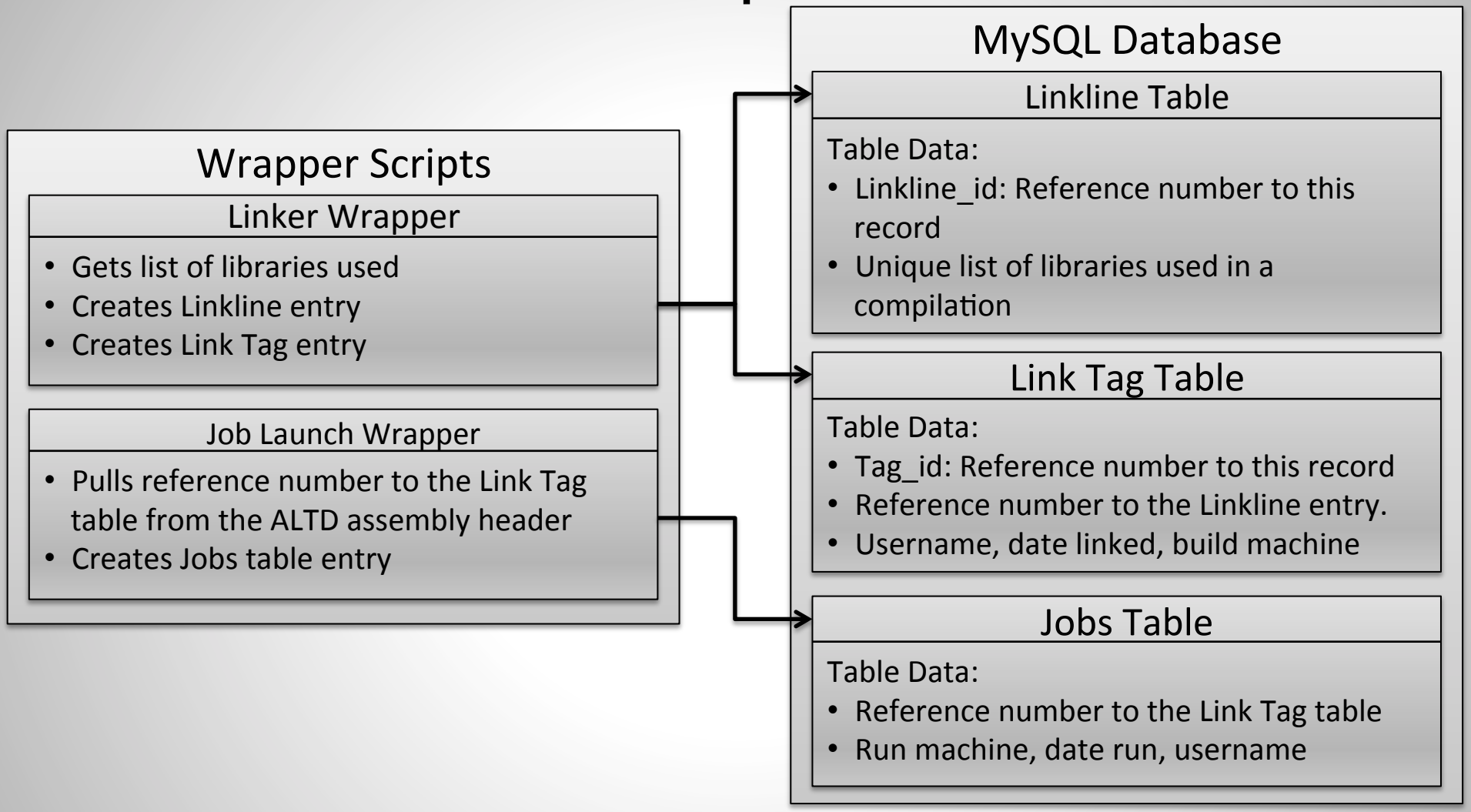
Linux Auditing Utility (auditd)

- System daemon
- Tracks specified files for access and or writes
- Stores to log files

Linux Auditing Utility (auditd)

- Provides log information on targeted files or directories
  - Object access and use
  - Stores to log files
  - Provides commands for searching log data
- Originally meant for security
- Root access required to add files to be tracked
- Log file is readable by a non-root linux group

ALTD Operation



ALTD Alterations

- Support added for wrapping mpirun and mvapich2 job launchers, while still allowing aprun
- Detects version of prefered MPI wrapped compiler and MPI executable and wraps accordingly
- Created script to query the database and collect the number of times a library has been compiled and run
- Made ALTD entries generic across servers

ALTD

Pros:

- Data is well organized into structured database
- Theoretical constant low overhead
- Tracks all libraries automatically

Cons:

- Requires a SQL database
- Tracks libraries used at compile time, not necessarily libraries used at run time

auditd

Pros:

- Standard Linux daemon
- Well tested by the Linux community
- Logs to a flat file

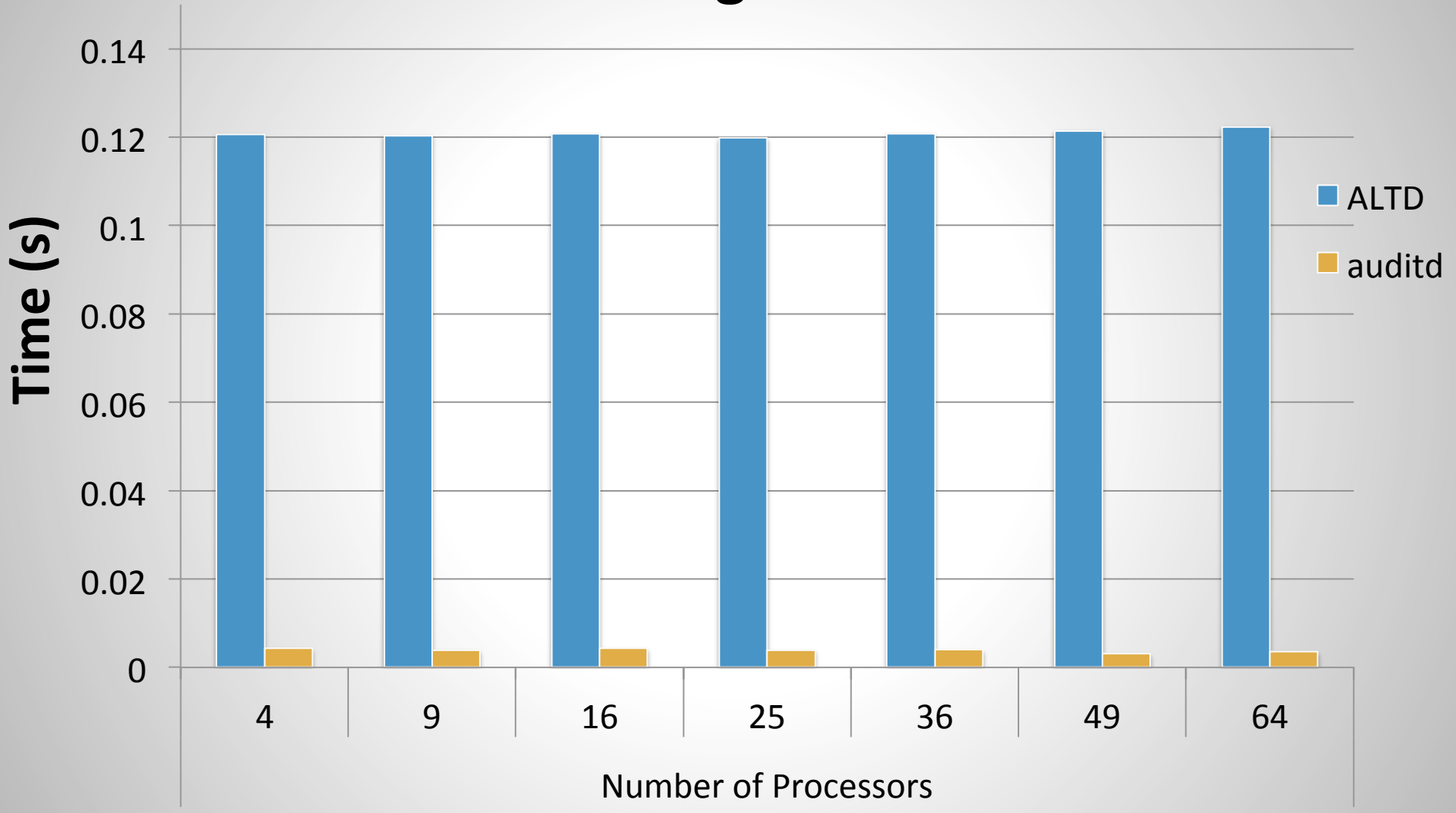
Cons:

- Dynamically linked libraries cannot be reliably tracked, the .so files are cached in memory and not read at every run
- Statically linked libraries can't be track at runtime as they are in the executable
- A single make can generate numerous log entries

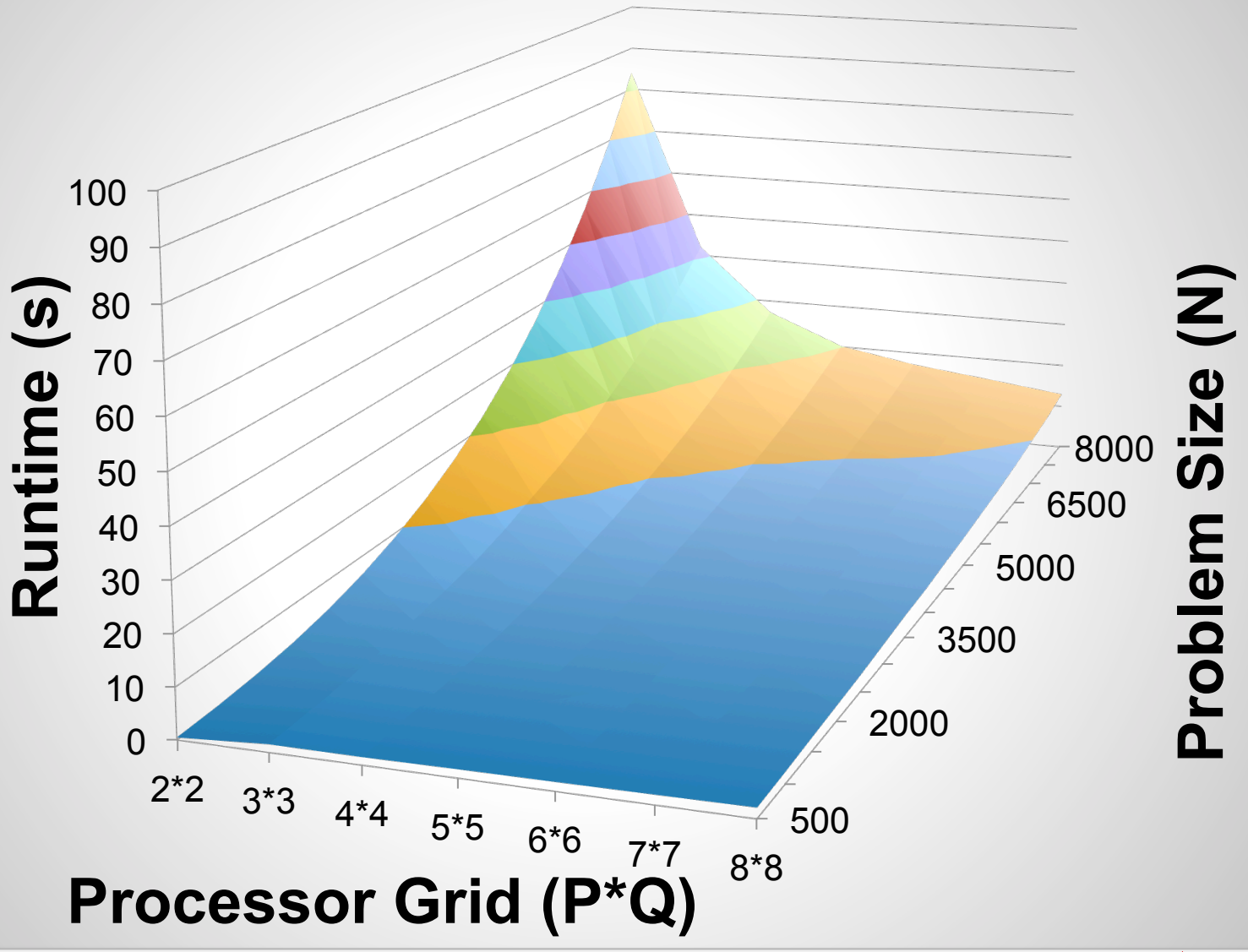
System Overhead Testing

- Interested in increase in compile and run time of different solutions over a control time
- Linpack was run while varying the problem size and the number of processors working on the problem
- A simple "do nothing" MPI program was also tested to investigate launch time more easily

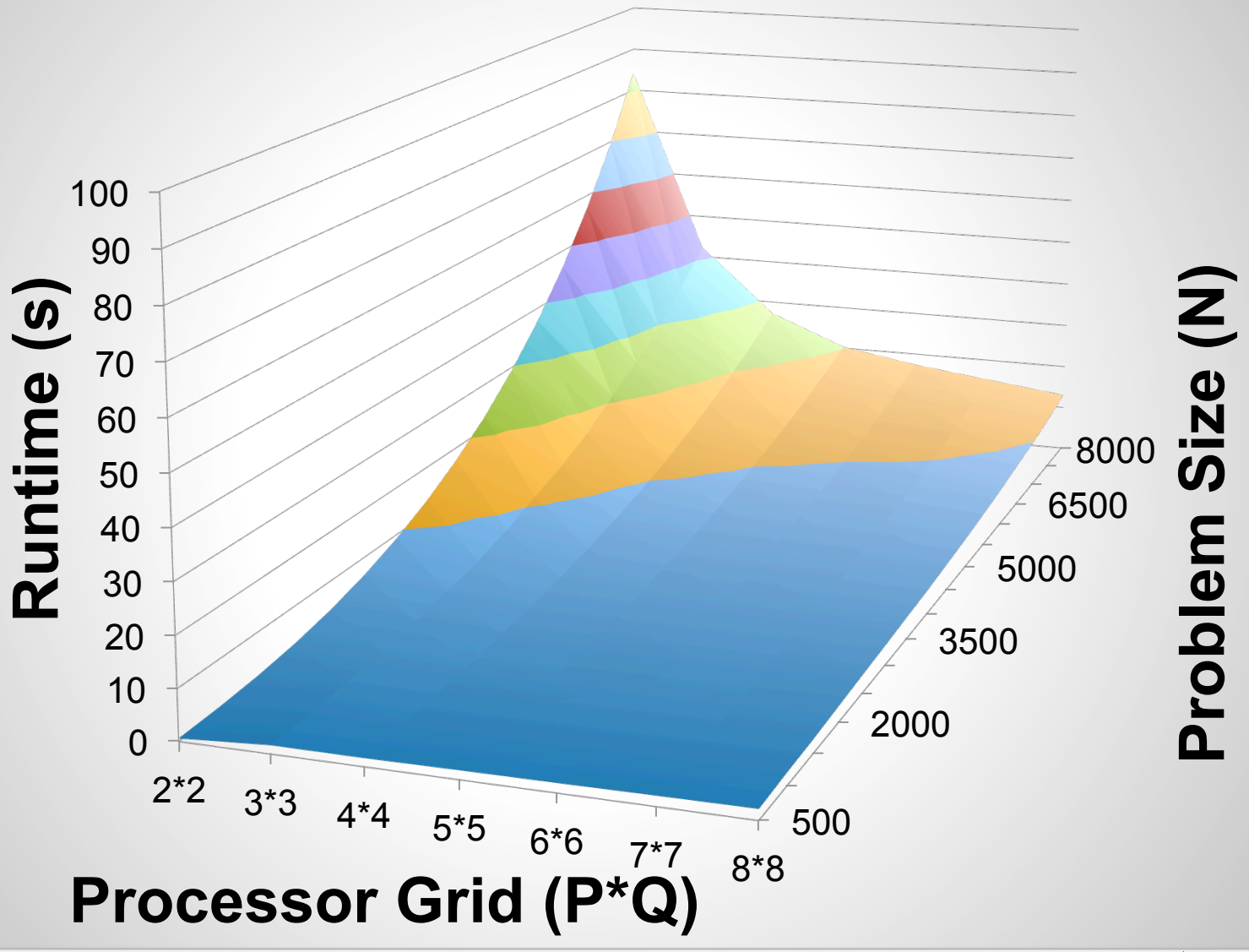
Additional Runtime for a Simple MPI Program



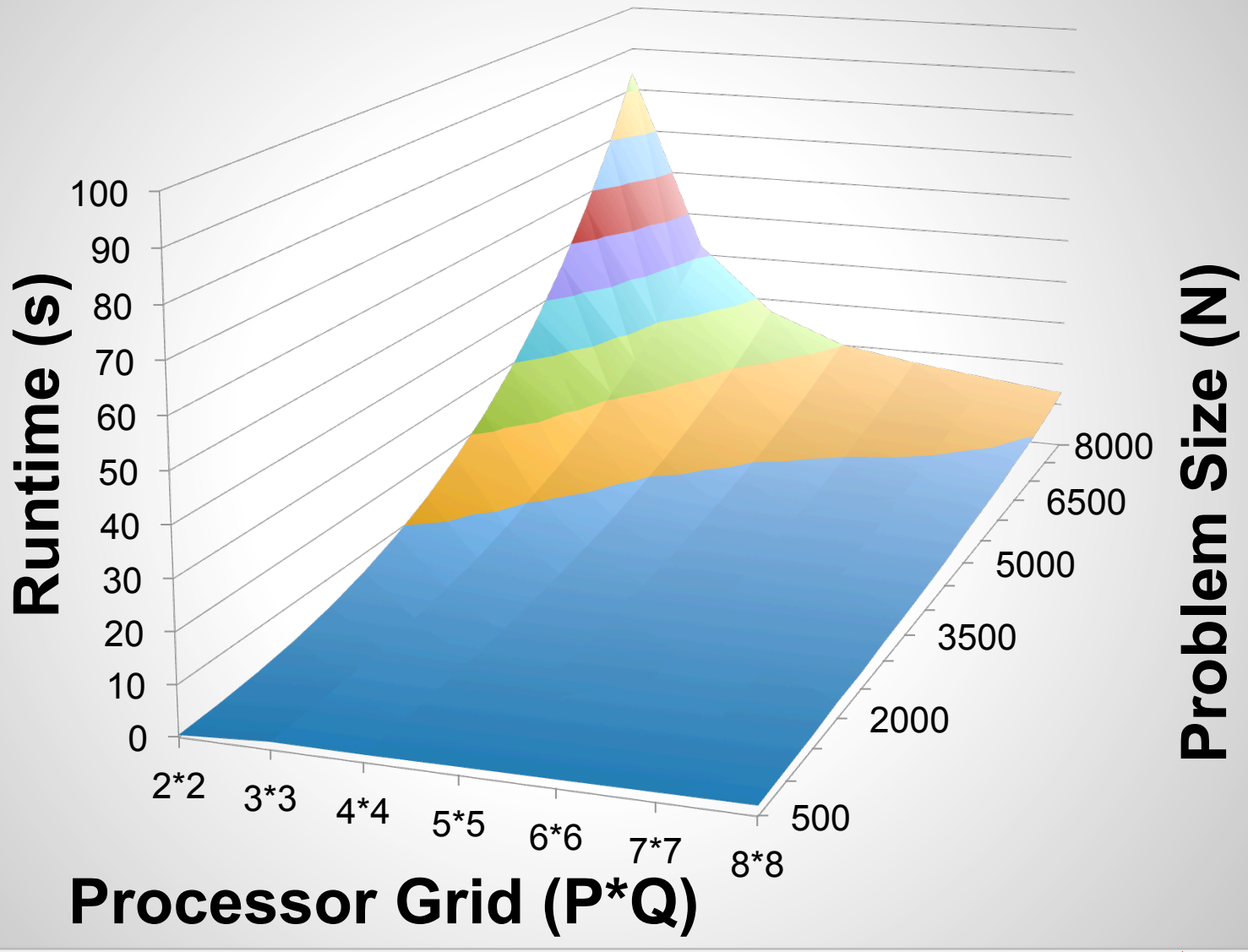
Linpack Runtime (Control)



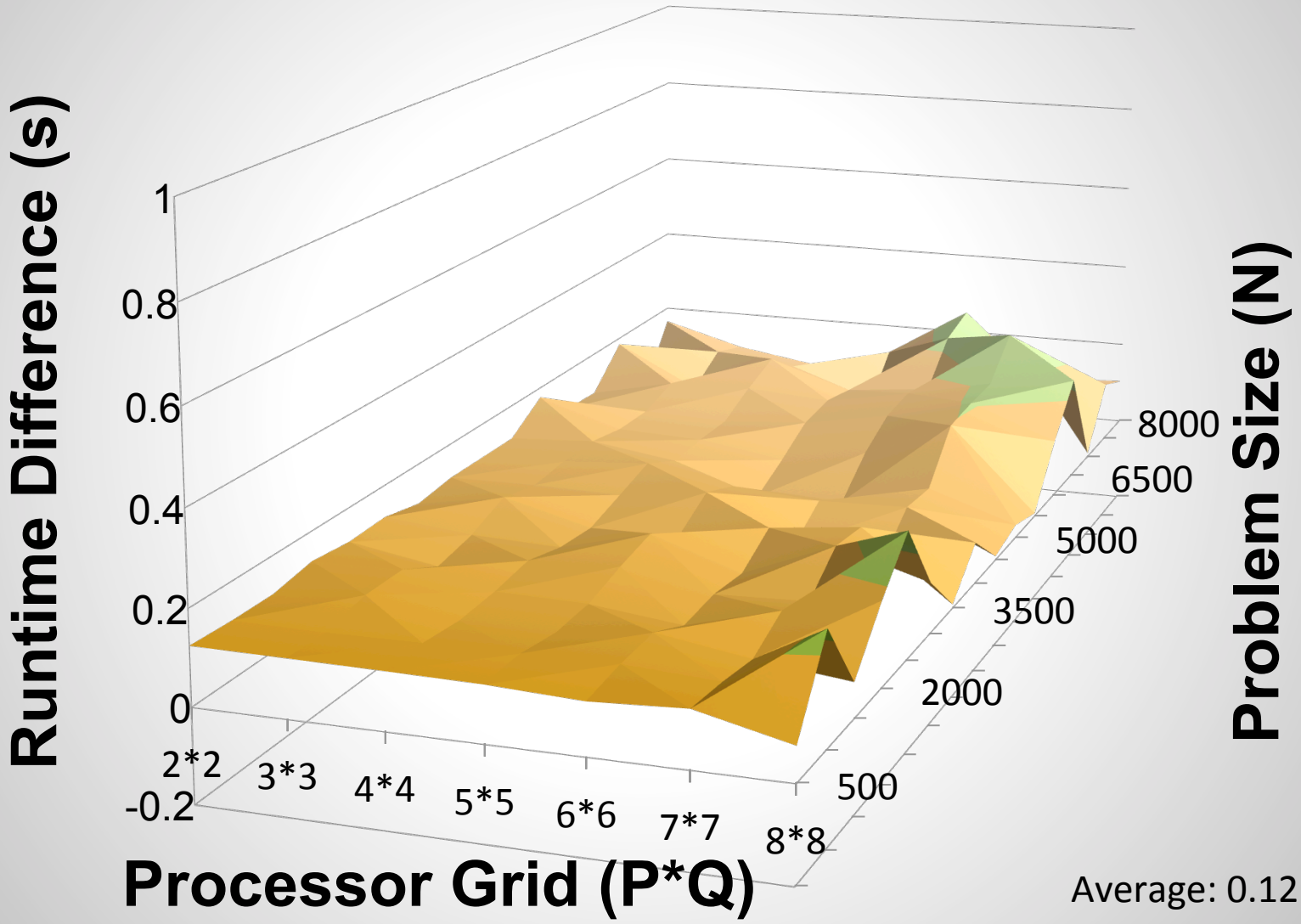
Linpack Runtime with ALTD



Linpack Runtime with auditd

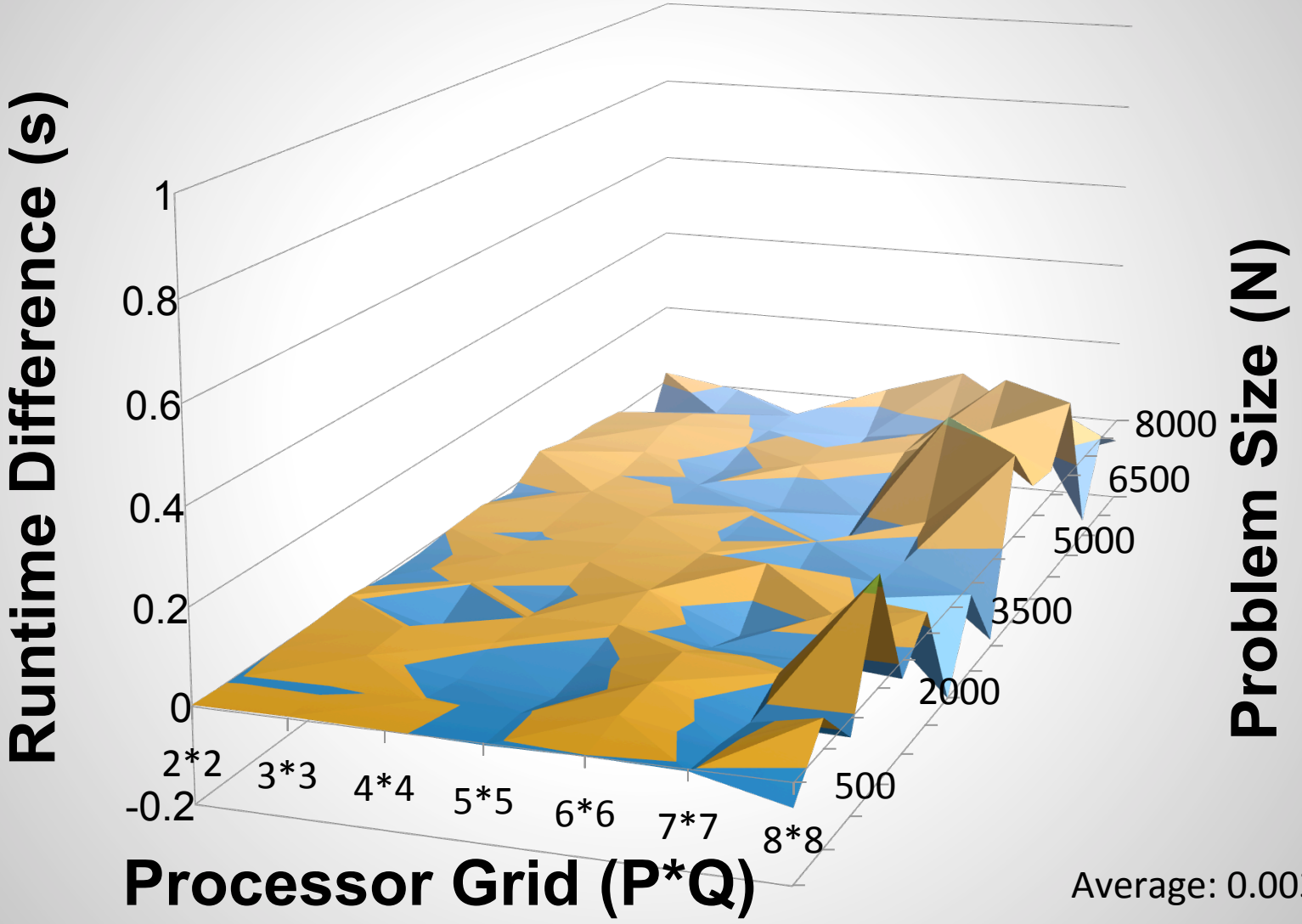


Additional Runtime for Linpack with ALTD



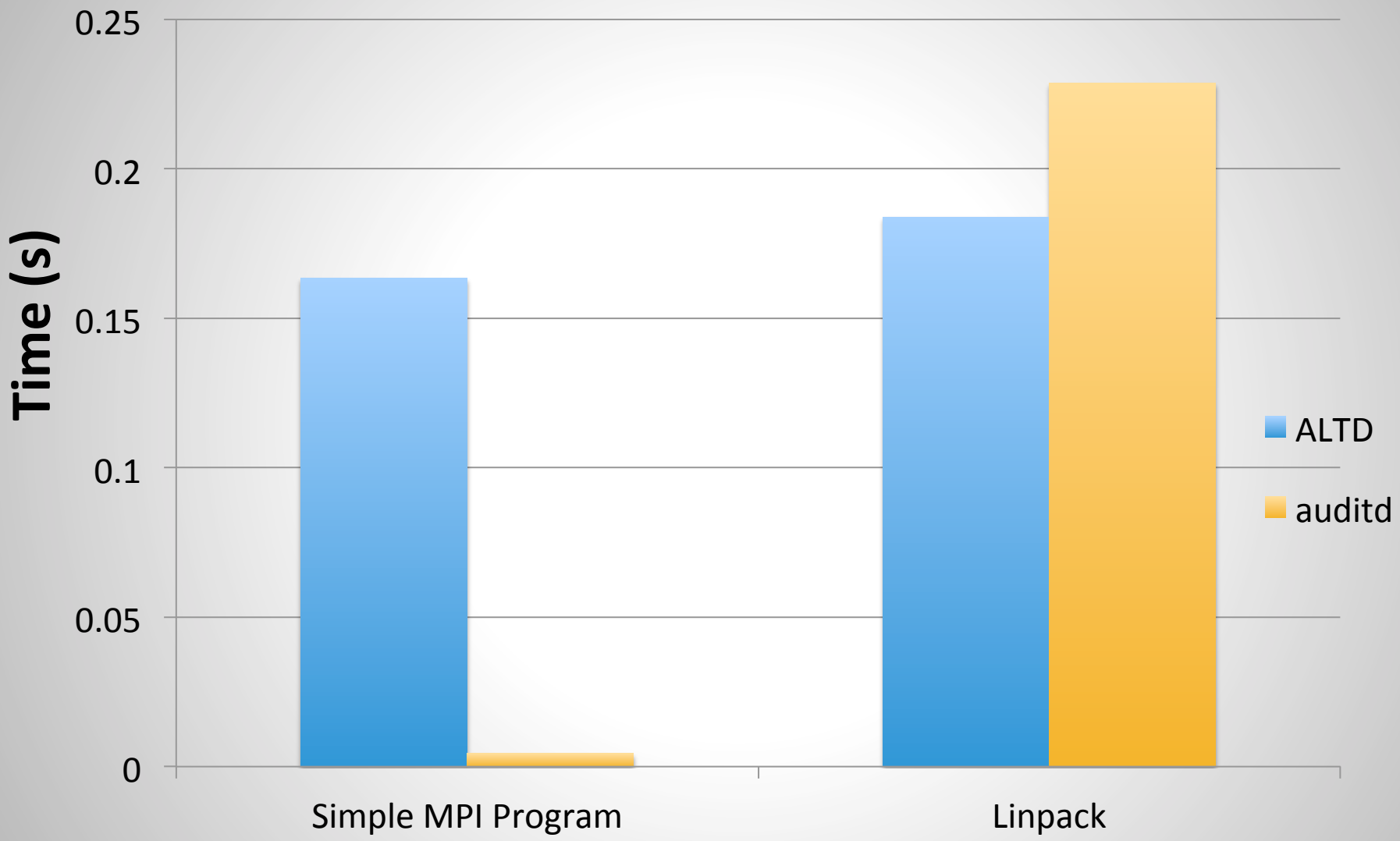
Average: 0.121 (s)

Additional Runtime for Linpack with auditd



Average: 0.00304 (s)

Additional Compilation Time Required



Summary

- ALTD
  - The Automatic Library Tracking Database has the potential to track libraries well in HPC
  - Performance overhead is minimal and seems to be constant, regardless of job size
  - Additions to the software have provided for simplicity in utilization and data evaluation.
- auditd
  - Outperforms ALTD in some cases but falls behind compiling larger programs
  - auditd is not sufficient to track all types of libraries during compilation and runtime

Thanks!

We would like to thank the following individuals for their support throughout the workshop:

Georgia Pedicini
David Gunter
Dane Gardner
Matt Broomfield
Carol Hogsett
Josephine Olivas
Carol Connor
Gary Grider