

# Towards Real-Time, Many Task Applications on Large Distributed Systems

- focusing on the implementation of RT-BOINC

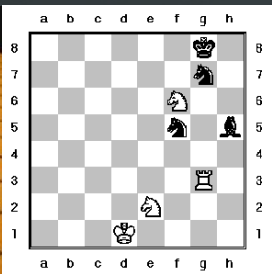
Sangho Yi (sangho.yi@inria.fr)



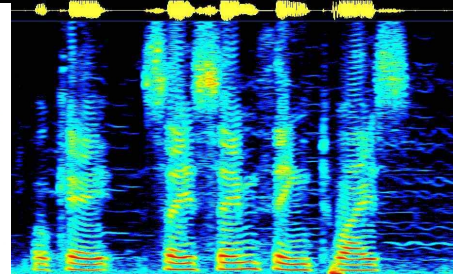
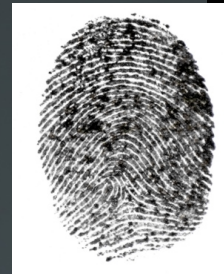
# Content

- Motivation and Background
- **RT-BOINC** in a nutshell
  - Internal structures
  - Design & implementation
- Conclusions and future work

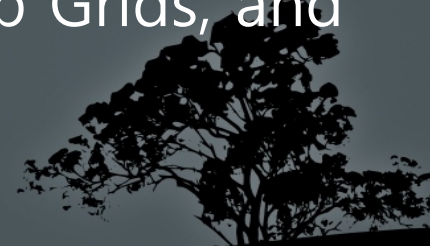




# Motivation



- Demands for computing large-scale real-time(RT) tasks increased in distributed computing environment
  - Chess, Game of Go
  - Real-time Forensic Analysis
  - Ultra HD-level Real-time Multimedia Processing
  - ...
- Lack of support for RT in existing Desktop Grids, and Volunteer Computing environment



# About BOINC



- BOINC is tailored for maximizing task throughput, not minimizing latency on the order of seconds.
  - Same as XtremWeb and Condor
- A BOINC project has
  - A BOINC server (web, storage, database, ...)
  - Multiple BOINC clients
  - Network connection between server - clients



# BOINC Projects



- Normally perform a few transactions in 1 sec with host clients.
  - 1~15 transactions in 1 sec (ref. <http://boincstats.com>)
- Send large chunk of computation to the host clients.
  - a couple of **hours**, or even **days** of computation
- Does not have RT guarantee
  - Because it is tailored for maximizing total amount of computation.

# Significant Gaps here...

- "I need a 10-second-car." - in the movie "Fast & Furious"

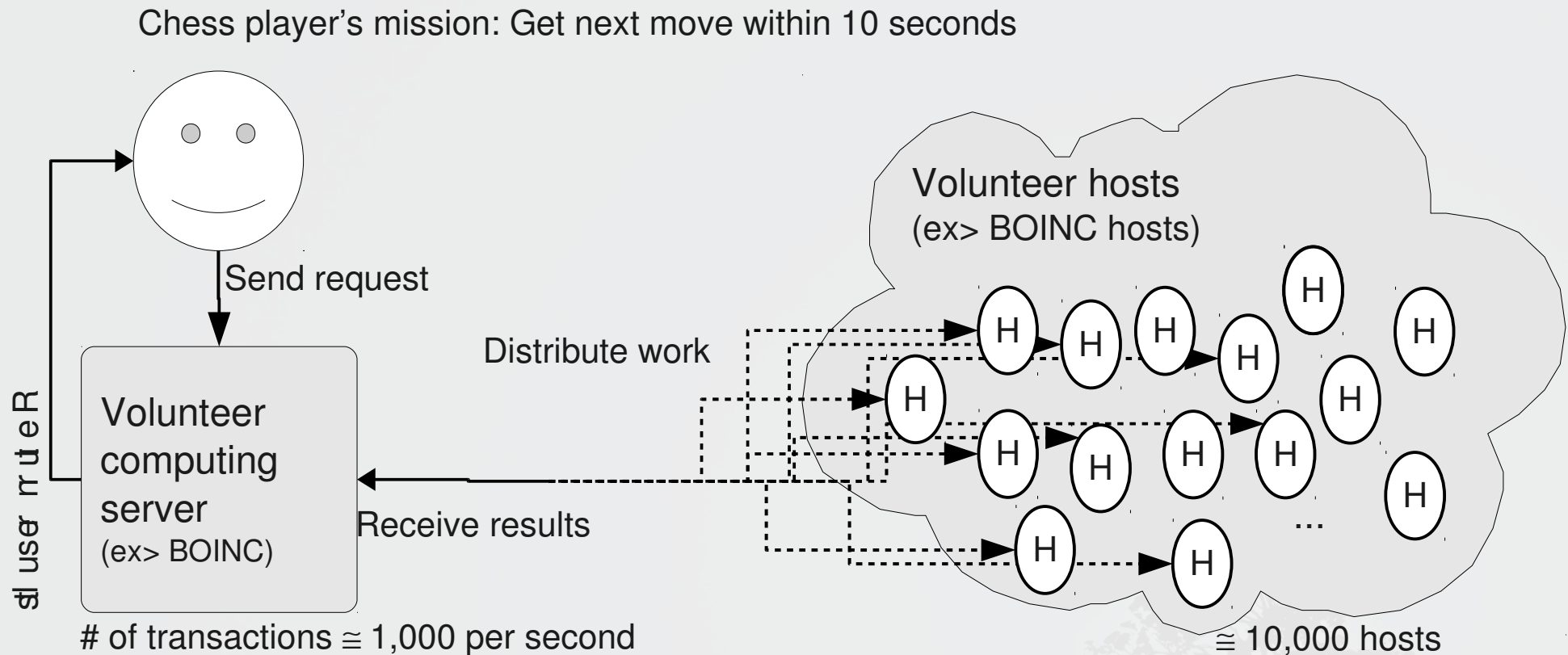


Vin diesel –  
the main actor in the movie



# Significant Gaps here...

- "We need a 10-second-completion." - in a "Chess game"



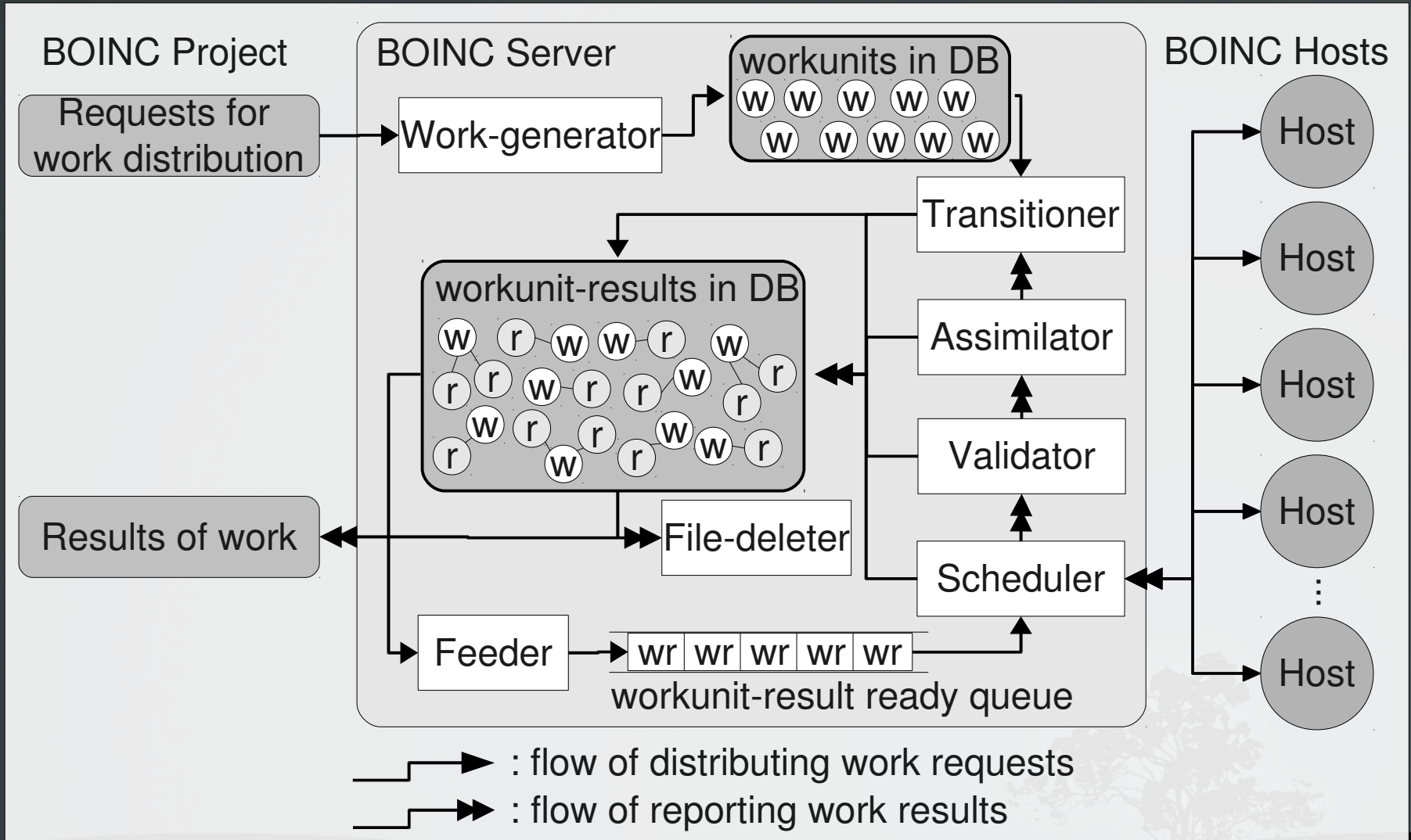
# RT-BOINC in a Nutshell

- RT-BOINC features
  - Providing low WCET (worst-case execution time) for all components
  - No database operations at run-time
  - $O(1)$  interfaces for data structures
  - Reduced complexity for server daemons
    - Almost  $O(1)$

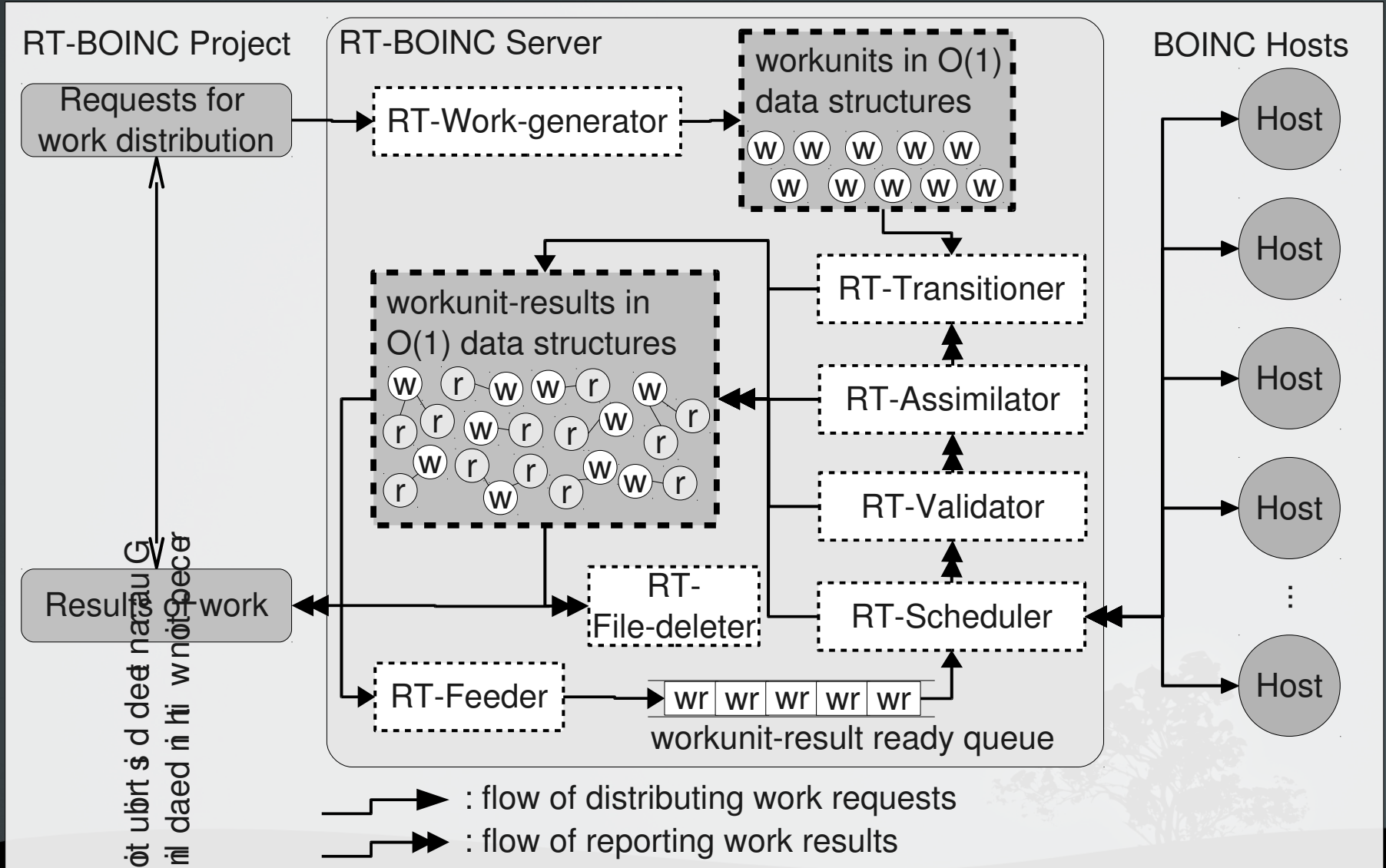




# Original BOINC Internal

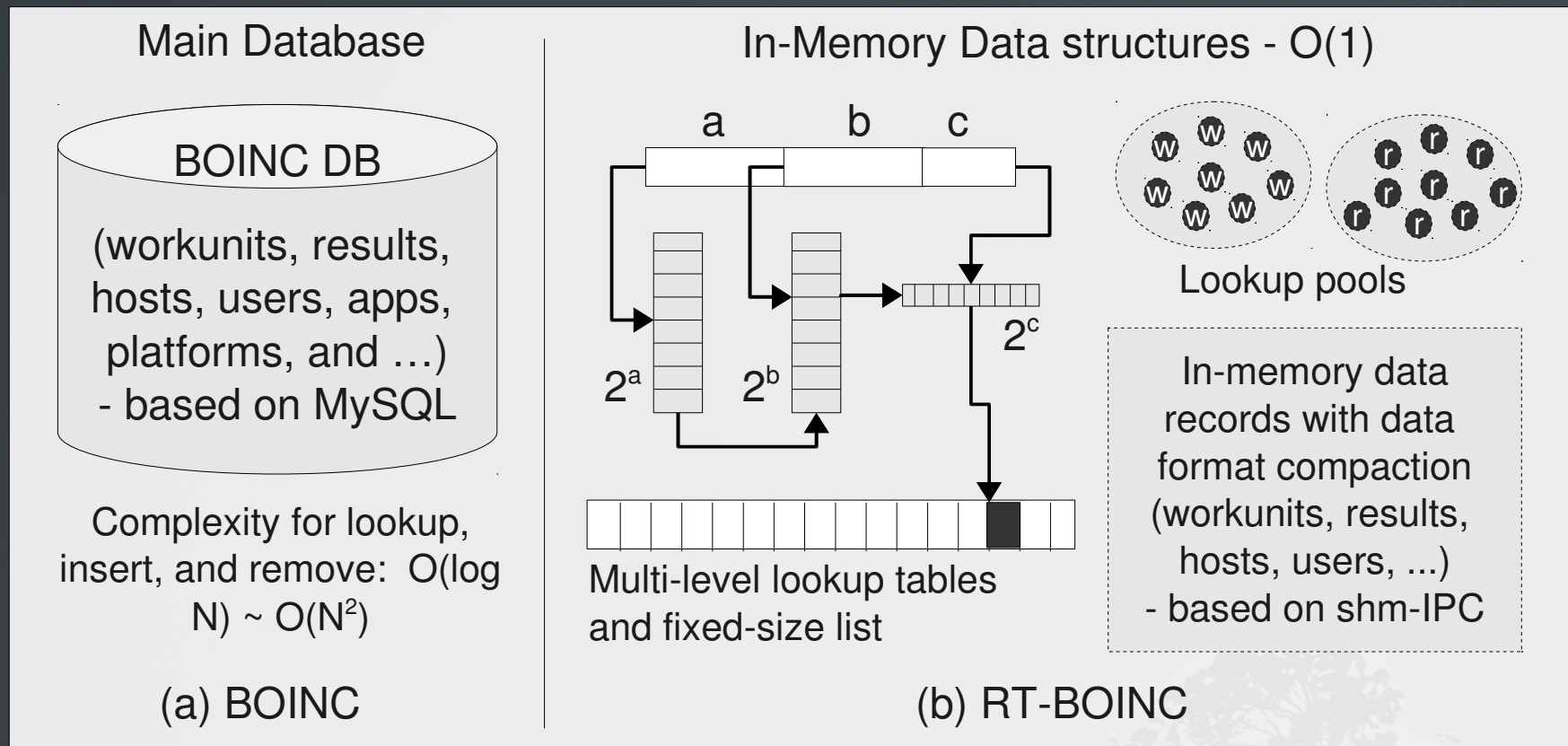


# RT-BOINC Internal



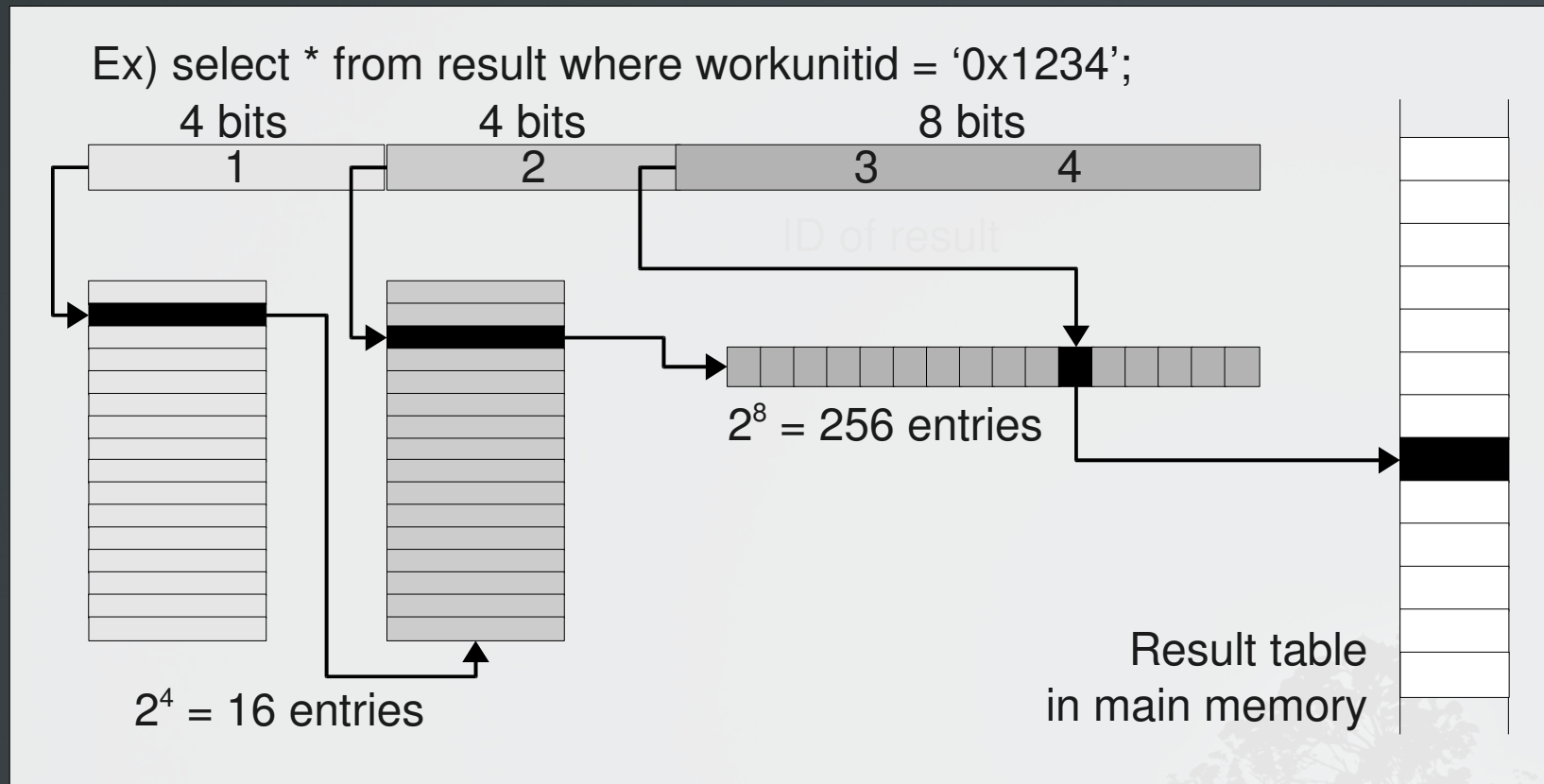
# Data management

- MySQL Database vs. In-memory data structures



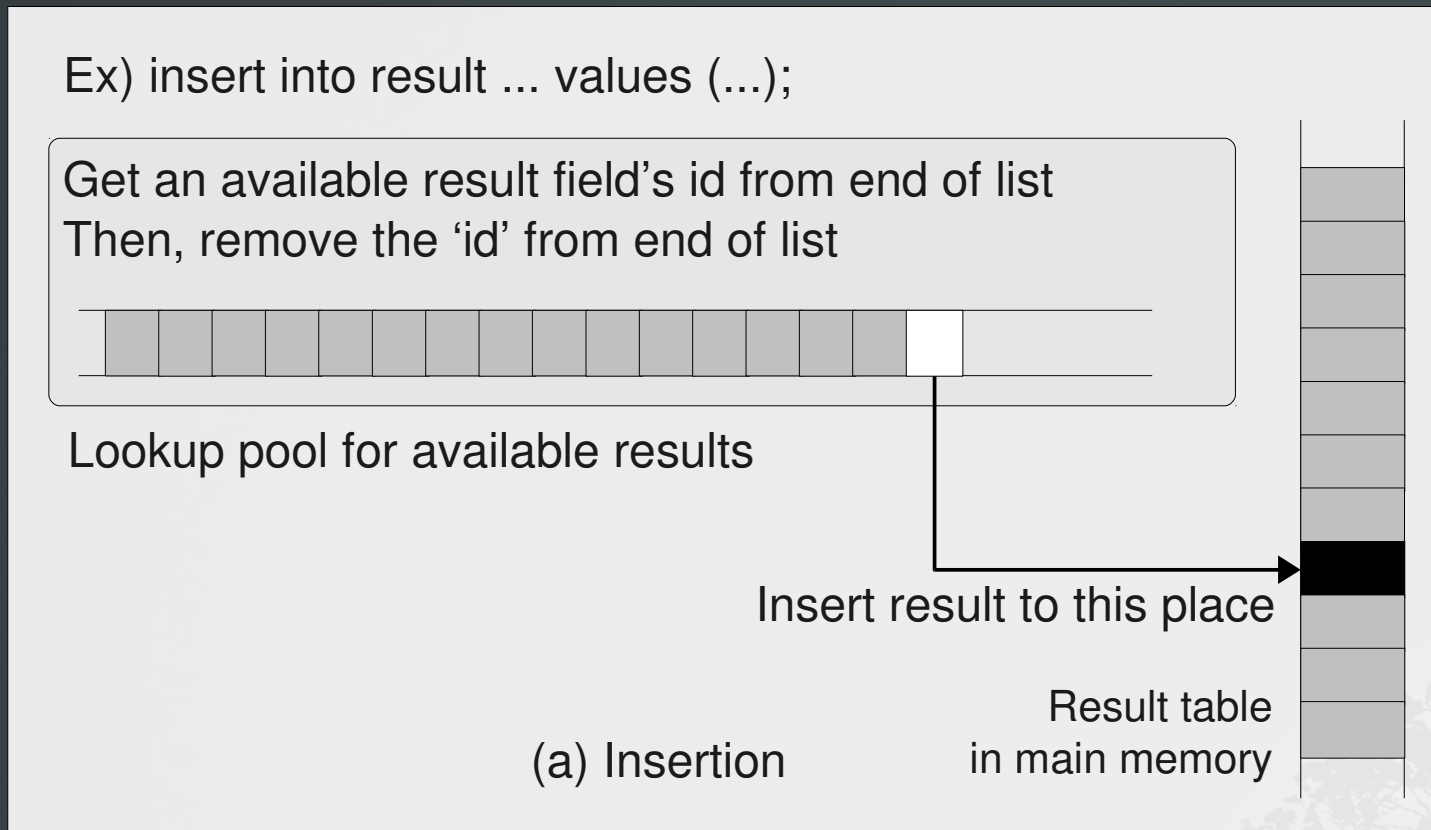
# Example 1) select from where;

- Retrieving RESULT from the O(1) data structure



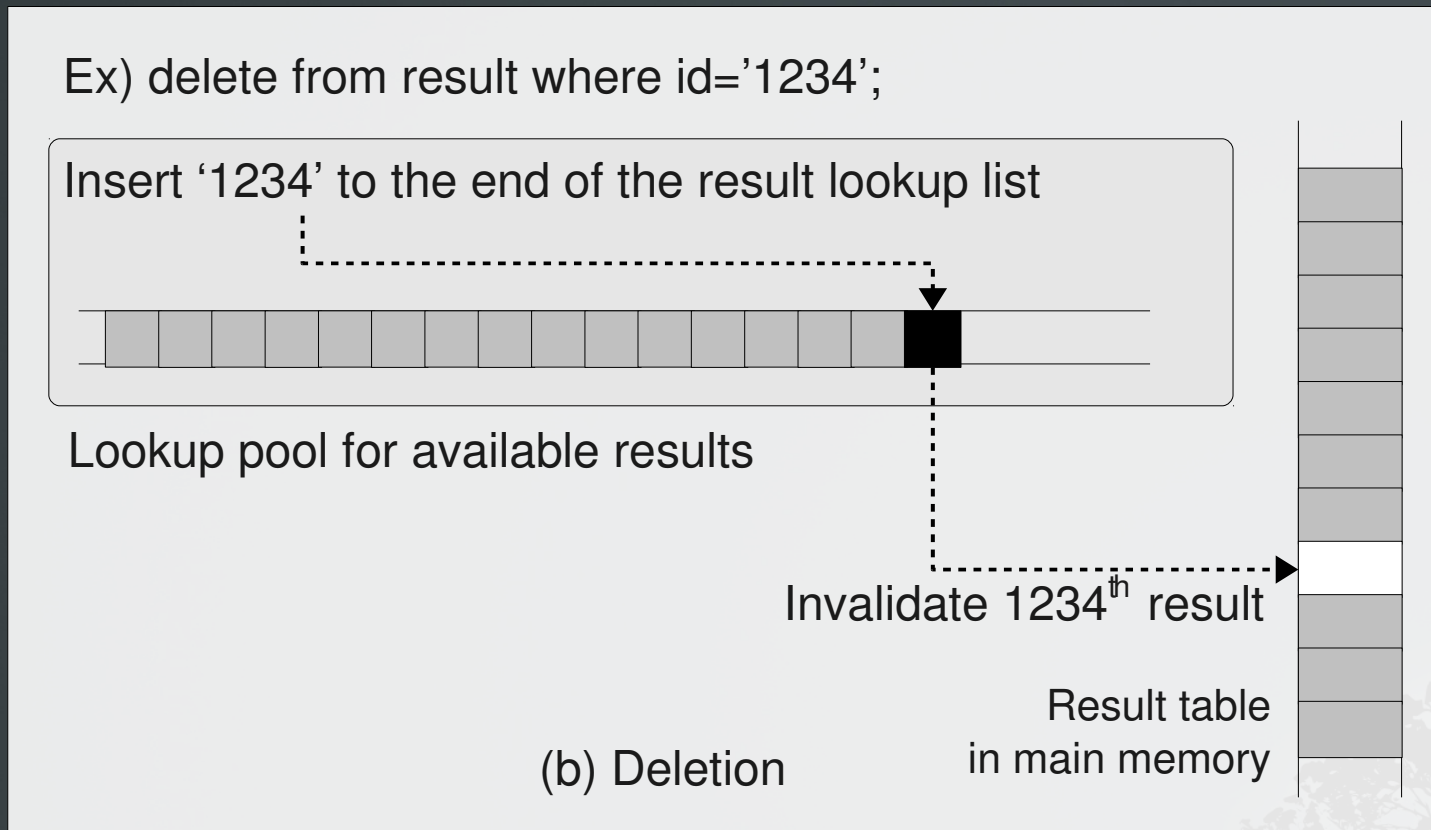
# Example 2) insert into values(...);

- Inserting RESULT to the O(1) data structure



# Example 3) delete from where;

- Deleting RESULT from the O(1) data structure



# Prototype Implementation

- Additional information
  - Compaction of BOINC's data format
  - Modification of PHP codes
  - Trade-offs between memory usage and WCET
    - Statically adjustable with parameters
  - Compatibility with BOINC
    - The rest parts are still compatible with BOINC.



# Size of Data Structures

- RT-BOINC uses the 'shared memory segment' IPC between server daemon processes to share the data structures.
- For 10,000 entries of hosts, results, workunits, it consumes totally **1.09GB** in main memory.
  - Memory overhead for O(1) data structures is **38.6%** of the total usage.
  - Using **1GB** memory is reasonable on the common-off-the-shelf 64-bit hardware platforms.





# Detailed information on the Web

- <http://rt-boinc.sourceforge.net>



**RT-BOINC** stands for a Real-Time BOINC

It was designed for managing highly-interactive, short-term, and massively-parallel real-time applications. We designed and implemented RT-BOINC on top of BOINC server source codes.

Contact information: [Sangho Yi](#) and [Derrick Kondo](#)

---

## For users

[Download RT-BOINC files](#)

[Project detail and discuss](#)

[Get support](#)

[Donate money](#)

## For developers

### Join this project:

To join this project, please contact the project administrators of this project, as shown on the [project summary page](#).

### Get the source code:

Source code for this project may be available as [downloads](#) or through one of the SCM repositories used by the project, as [page](#).

---

## About RT-BOINC

# Source code on the Web

- <http://sourceforge.net/projects/rt-boinc>

The screenshot shows the SourceForge project page for RT-BOINC. The page header includes the SourceForge logo, navigation links (Find Software, Develop, Create Project, Blog, Site Support, About), a search bar, and user information (Welcome, Sangho Yi (ANTIROOT), Log Out, Account). The project title is "RT-BOINC Beta by antiroot". Below the title are tabs for Summary, Files, Support, and Develop. The main content area contains a description of RT-BOINC, a "Download Now!" button for "rt-boinc.tar.gz (49.0 MB)", and a "View all files" button. There are also links for "WWW" and "TAGS". The "Features" section lists four items: Real-time server-side work unit transaction, Constant-time In-memory data structures (without using MySQL DB at rt), Using shared-memory IPC for both daemon processes (written in C) and, and Compatibility with the original BOINC. On the right side, there is a "Rate and Review" section with a thumbs up/down icon and a text box for an optional review. The review text reads: "30~100 Times higher performance than BOINC. 300~1000 Times lower WCET(worst-case execution time) for the given maximum load. Fantastic! Isn't it? lol".

SourceForge.net > Find Software > RT-BOINC

EDIT

REAL-TIME BOINC RT-BOINC Beta by antiroot

Share More Donate

Summary | Files | Support | Develop

RT-BOINC stands for a Real-Time BOINC. It was designed for managing highly-interactive, short-term, and massively-parallel real-time applications. We implemented RT-BOINC on top of the recent BOINC server source codes.

Download Now! rt-boinc.tar.gz (49.0 MB) OR View all files

WWW <http://rt-boinc.sourceforge.net>

TAGS [boinc](#) [gridcomputing](#) [real-time](#) [rt-boinc](#) [rtbnc](#) [volunteercomputing](#)

Features:

- Real-time server-side work unit transaction
- Constant-time In-memory data structures (without using MySQL DB at rt)
- Using shared-memory IPC for both daemon processes (written in C) and
- Compatibility with the original BOINC

Rate and Review

You gave this project a thumbs up!

or

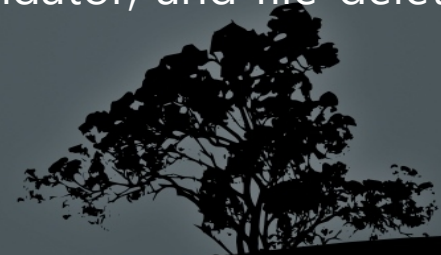
Add an optional review:

30~100 Times higher performance than BOINC. 300~1000 Times lower WCET(worst-case execution time) for the given maximum load. Fantastic! Isn't it? lol

Update Review or Delete Review

# Performance Evaluation

- Purpose: to measure real-time performance of **BOINC** and **RT-BOINC**
- Criteria: the worst-case and the average execution time
- Method: micro and macro benchmarks
  - Micro-benchmark: for each primary operation related to server process
  - Macro-benchmark: for each server process (including feeder, scheduler, transitioner, work-generator, assimilator, validator, and file-deleter)



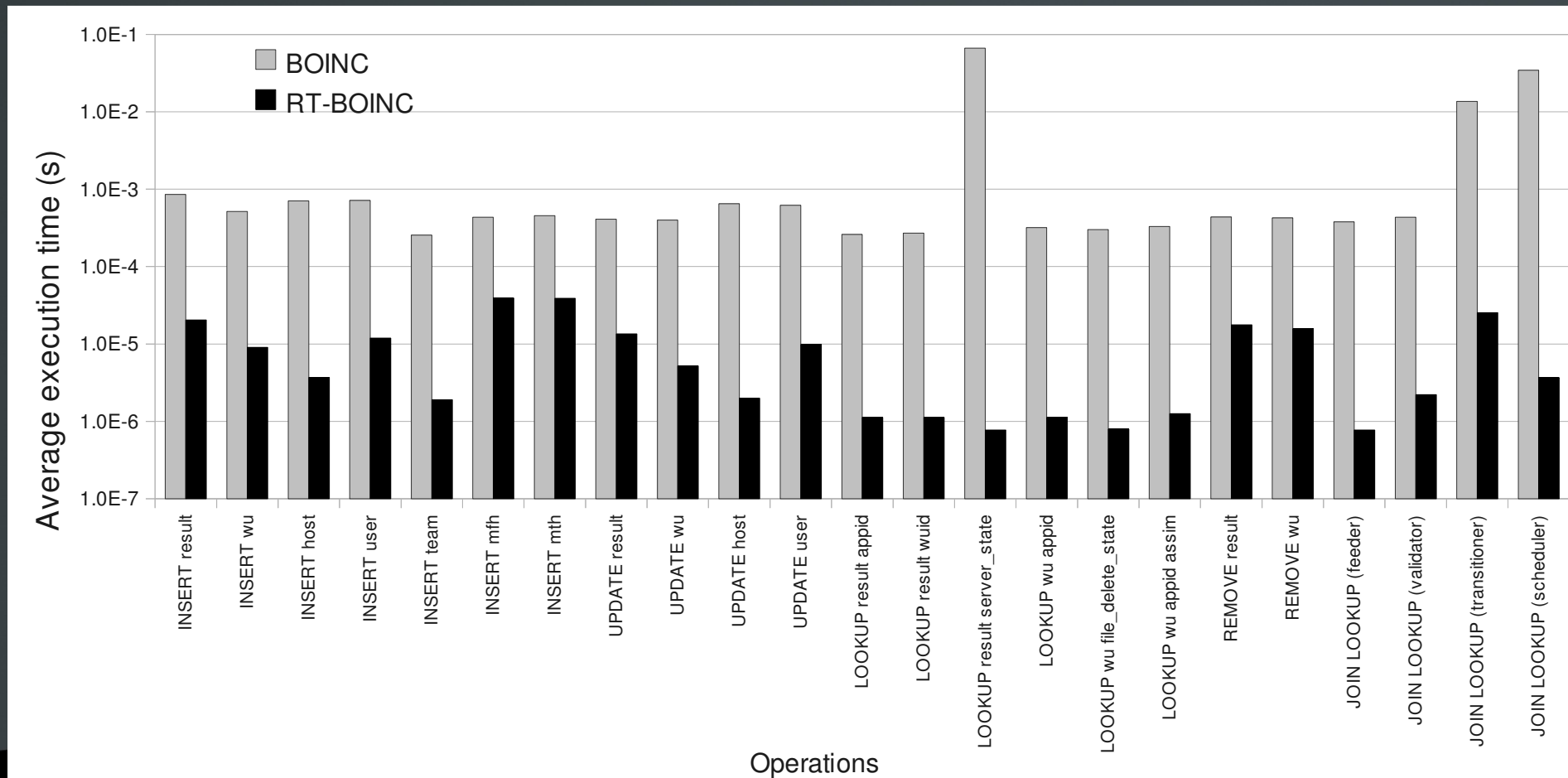
# Experimental Environment

- We used a little bit slow, common-off-the-shelf system. ;-)
  - For ease of reproduction of the results

Component	Description	Notes
Processor	1.60GHz, 3MB L2 cache	Intel Core 2 Duo
Main Memory	3GB (800 Mhz)	Dual-channel DDR3
Secondary Storage	Solid State Drive	SLC Type
Operating System	Ubuntu 9.10 (karmic)	Linux Kernel 2.6.31-19
BOINC version	Server stable version	Nov. 11, 2009 (from SVN)

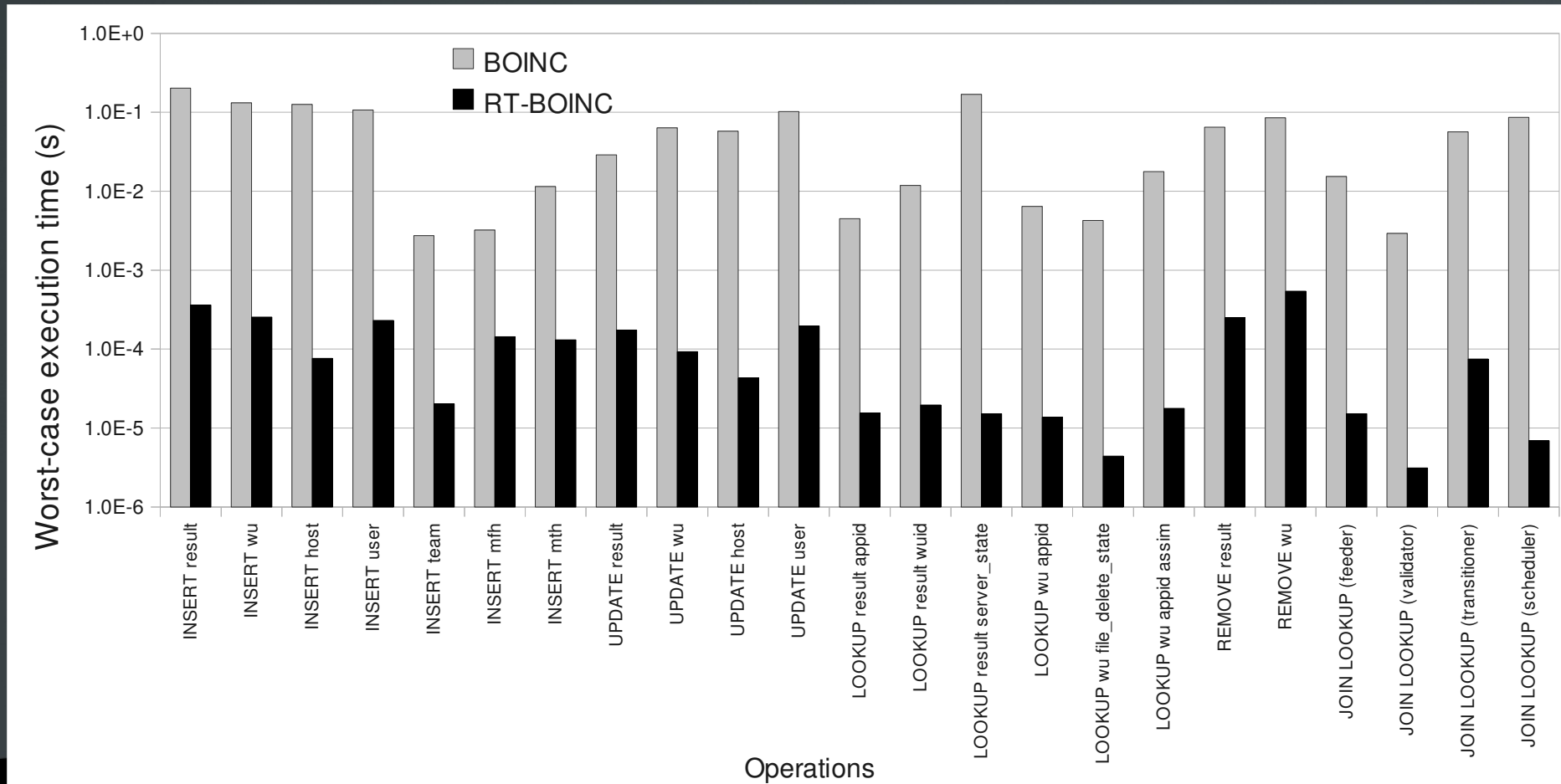
# Micro-benchmarks

- Average execution time (in seconds)



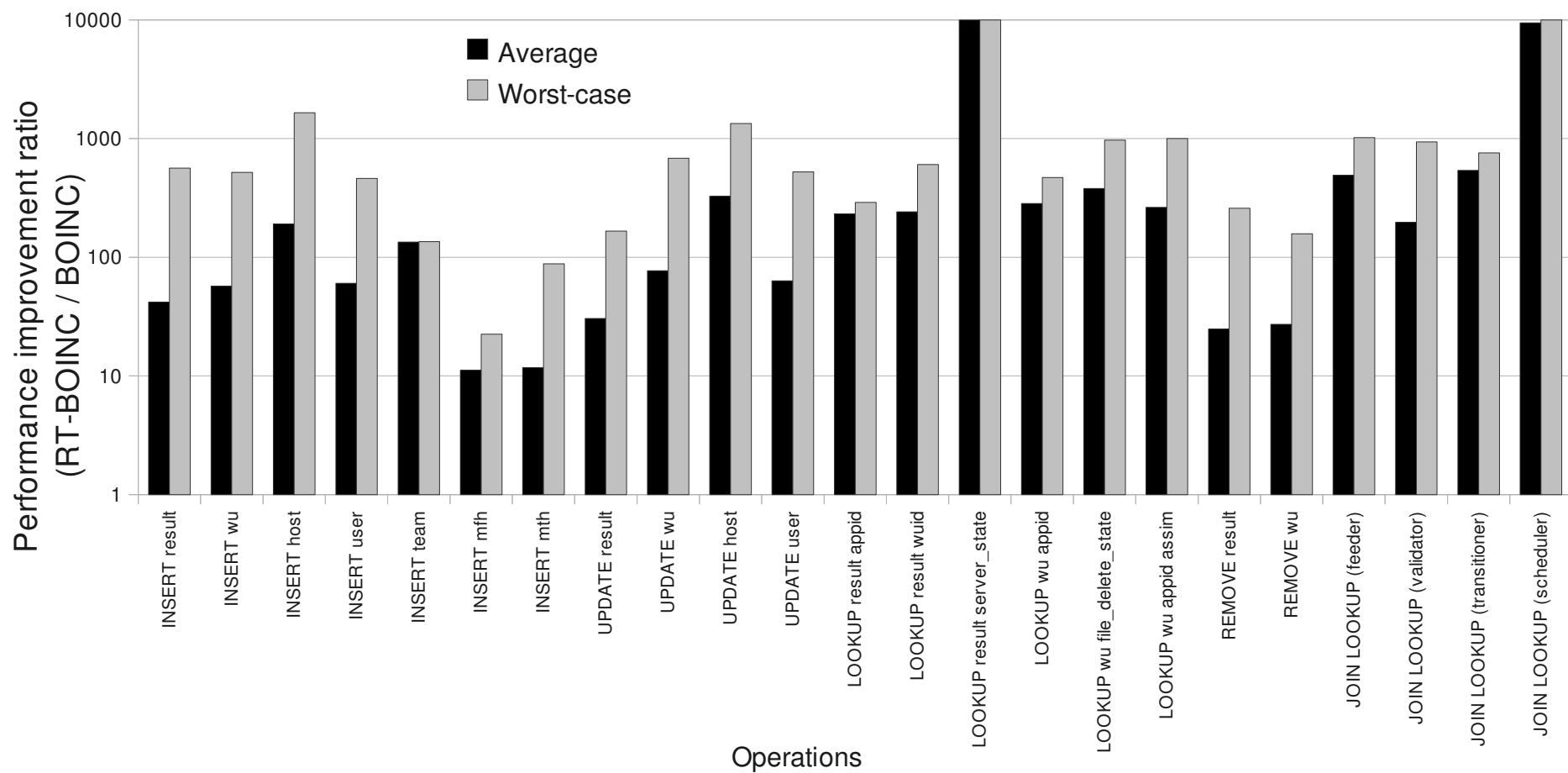
# Micro-benchmarks

- Worst-case execution time (in seconds)



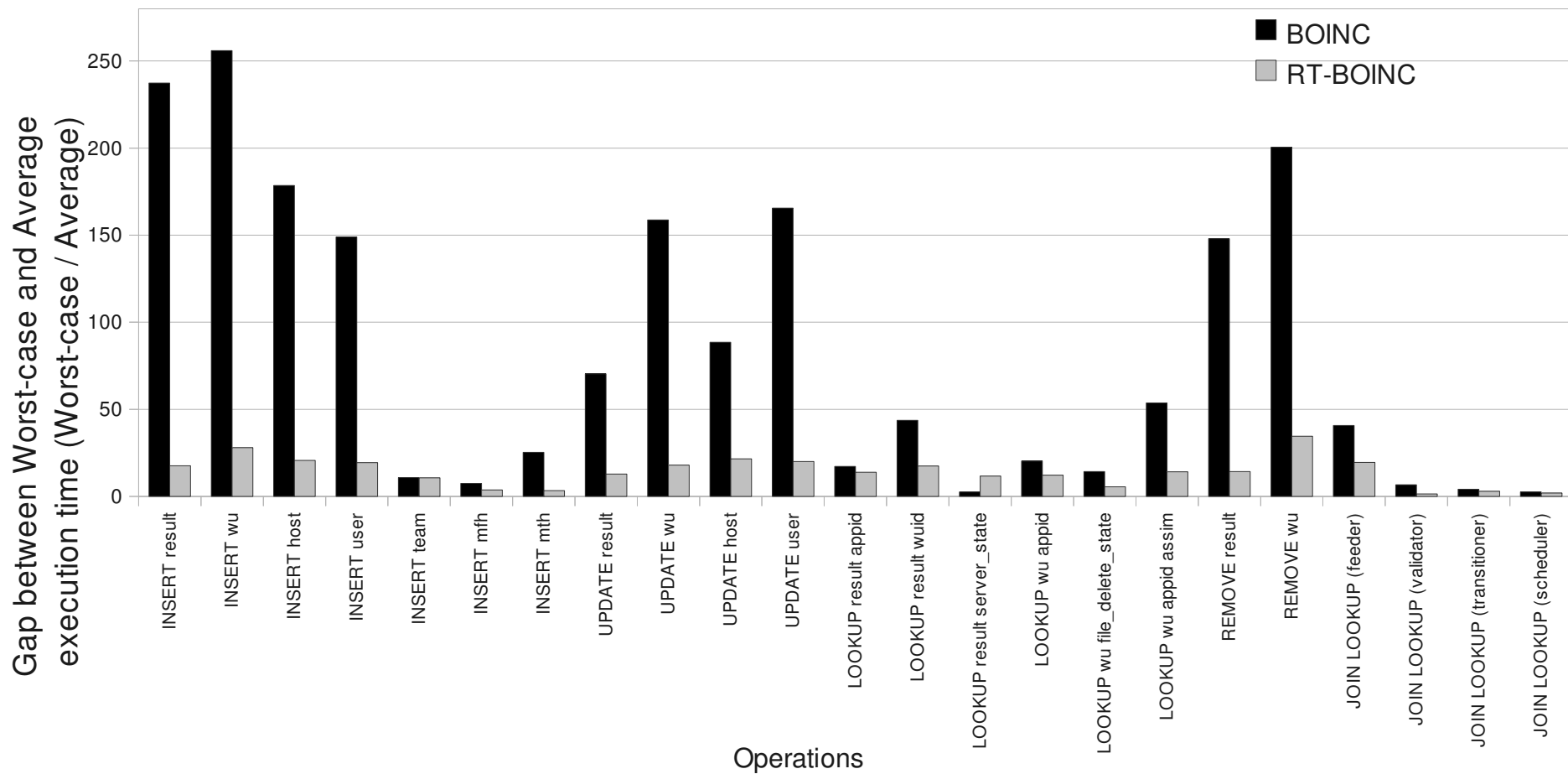
# Micro-benchmarks

- Performance improvement ratio (RT-BOINC / BOINC)



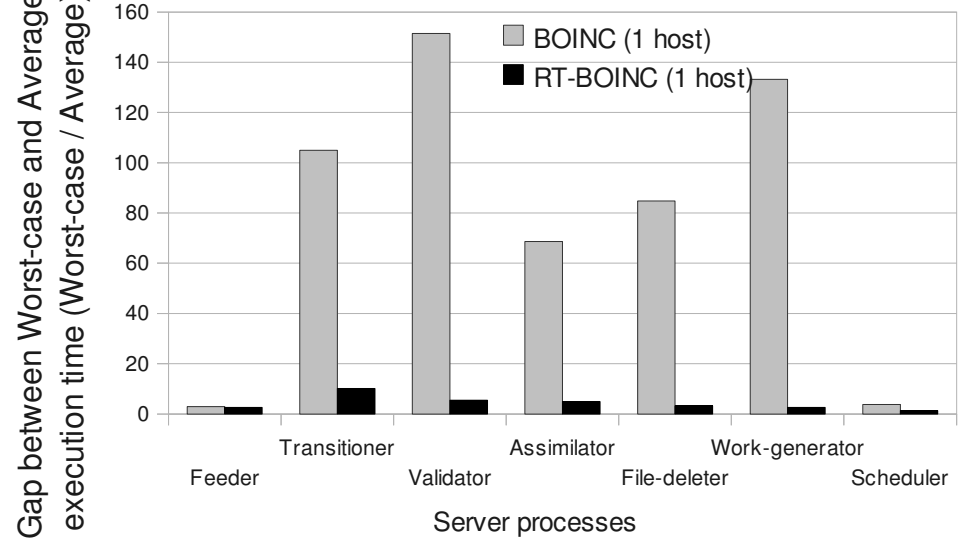
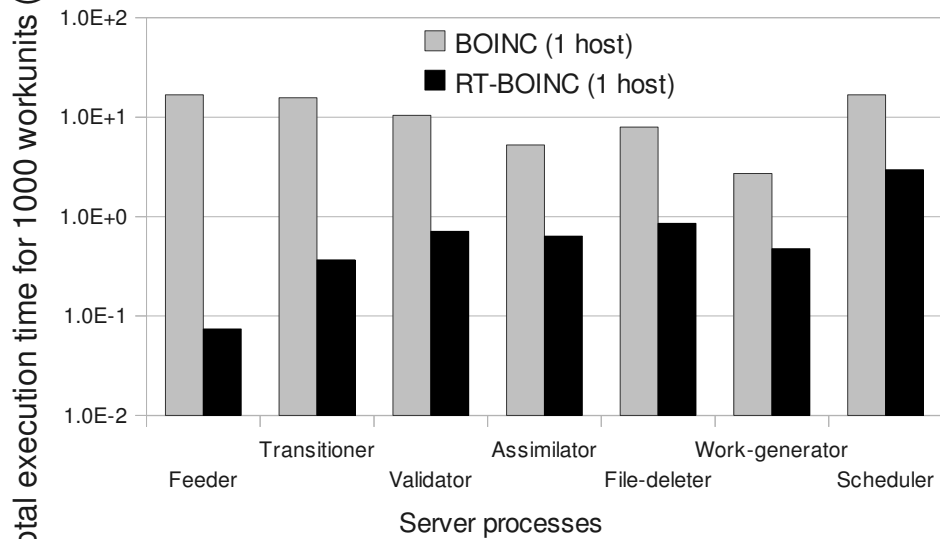
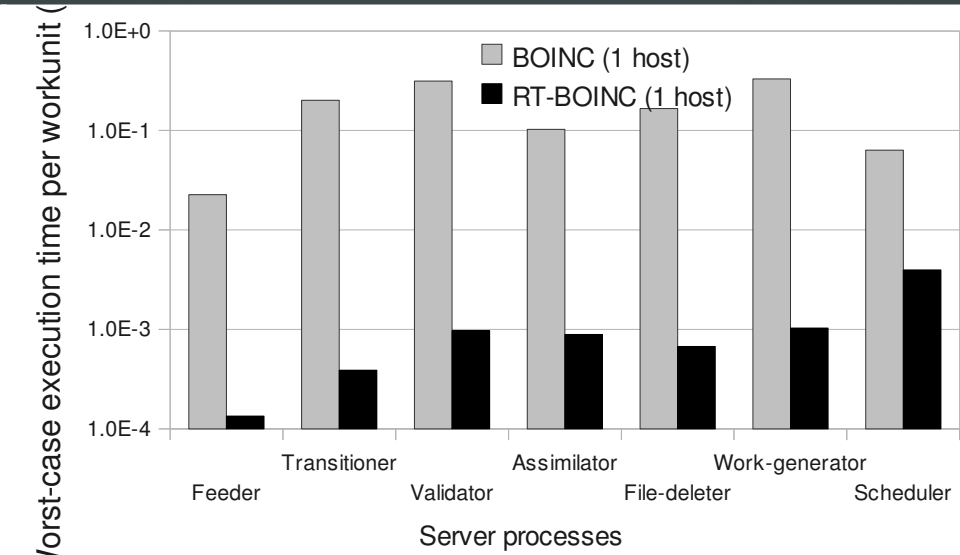
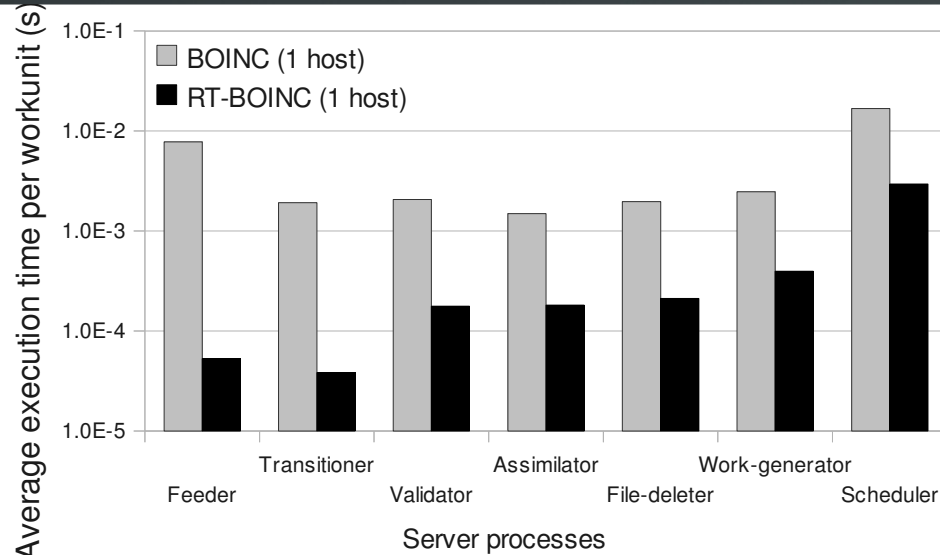
# Micro-benchmarks

- Performance gap between worst-case and average

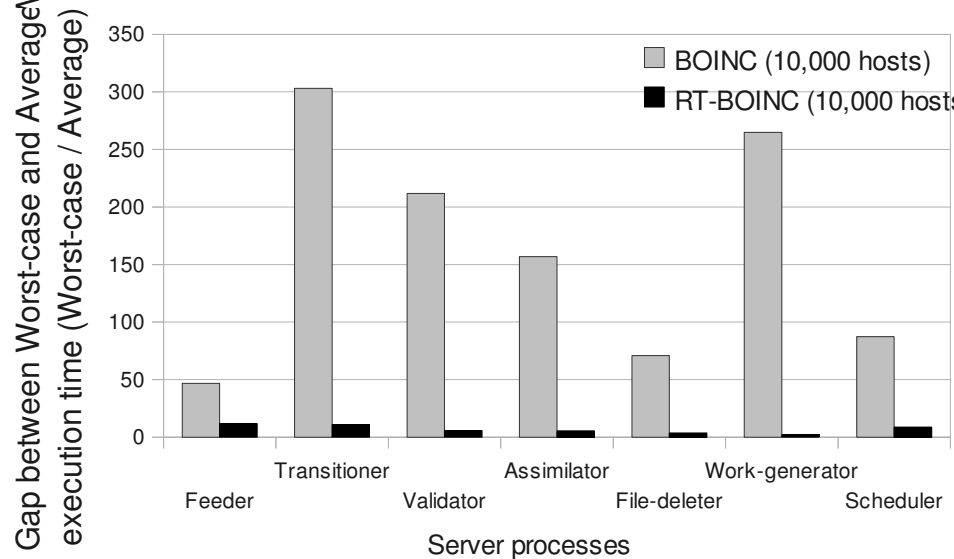
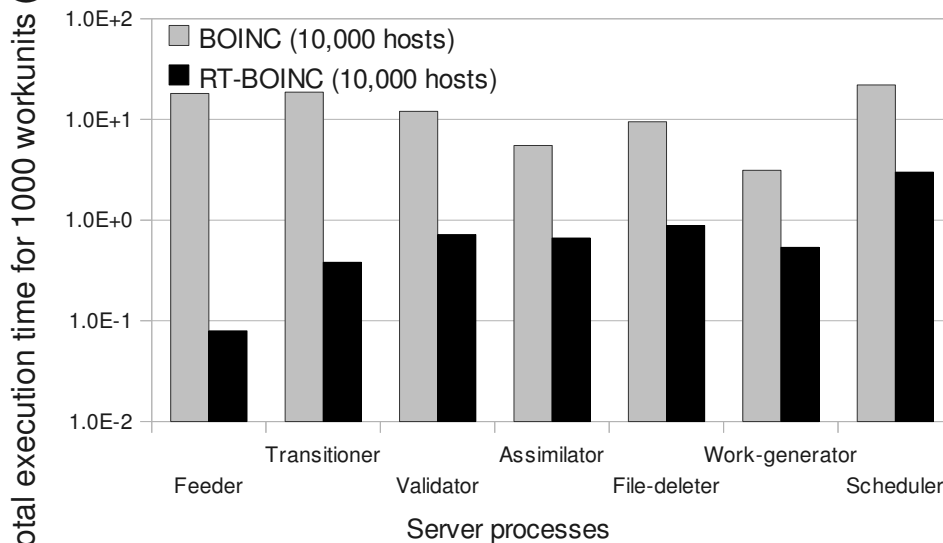
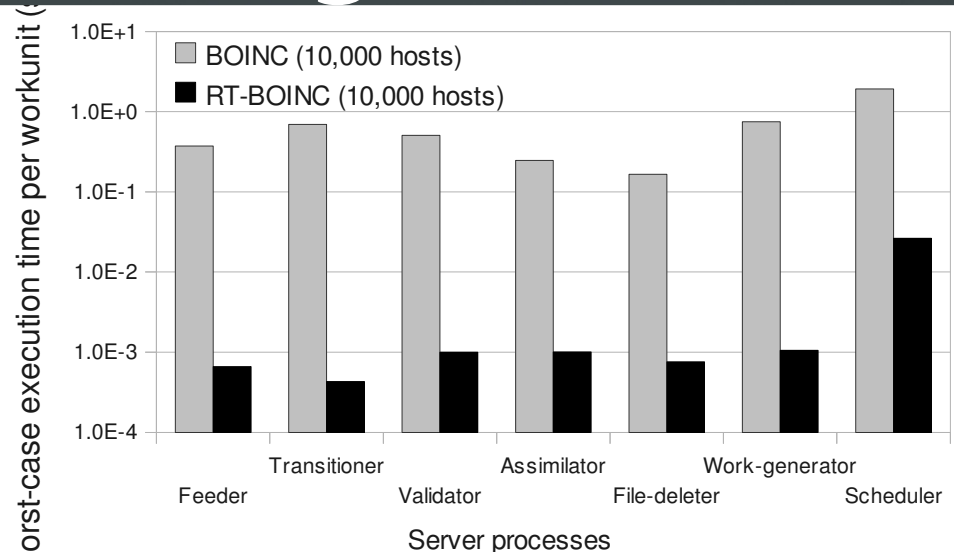
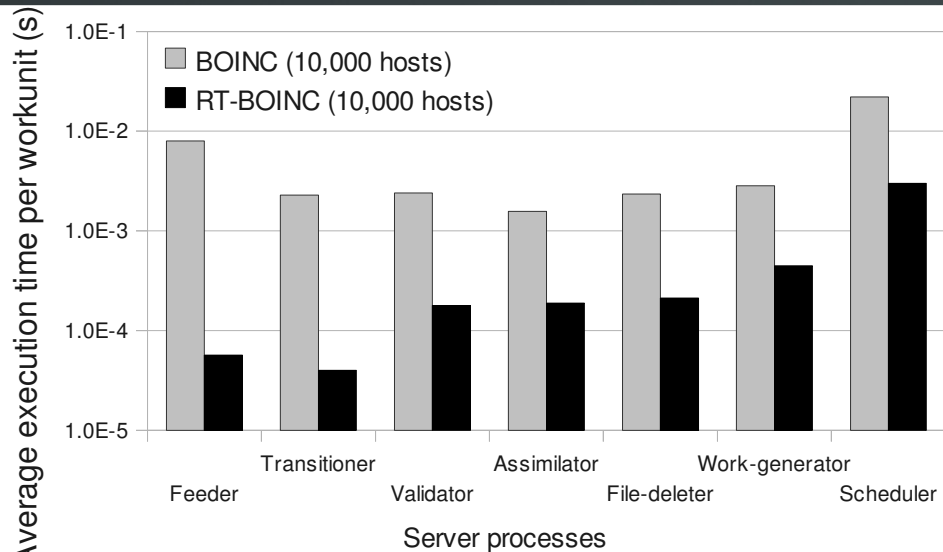




# Macro-benchmarks (low load)

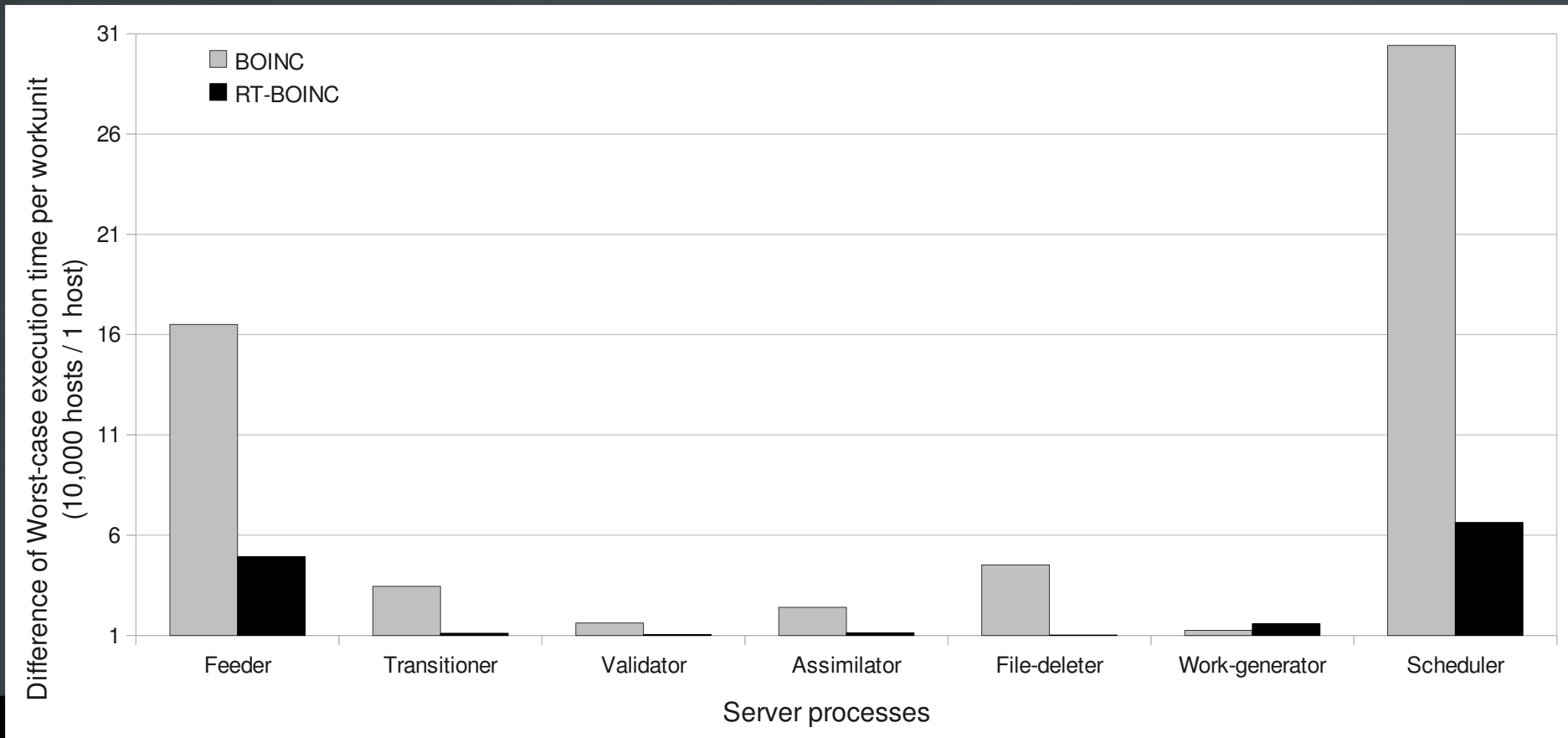


# Macro-benchmarks (high load)



# Macro-benchmarks

- Difference of worst-case performance between low and high load condition

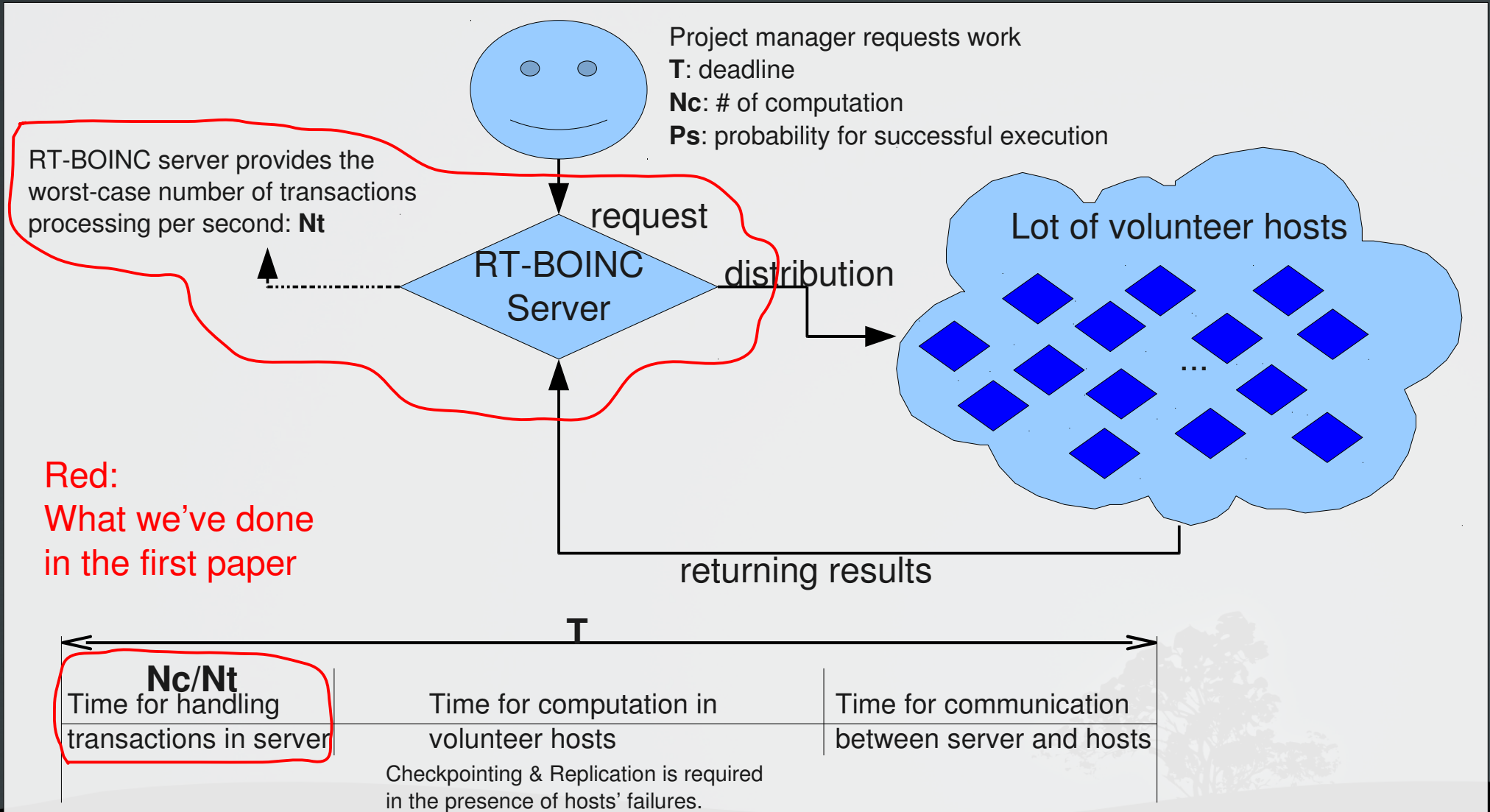


# Conclusions


- RT-BOINC provides...
  - 30~100 times higher average performance than BOINC.
  - 300~1,000 times lower WCET(worst-case execution time) for the given load condition.
  - less difference between the average and the worst-case performance.
  - less difference between low and high load conditions.



# Future work (The rest part)



# Future work (Remaining issues)

- Providing 'dynamic shared-memory management' for reducing memory usage
  - Studying trade-offs between time(WCET) and space(memory usage)
  - Providing 'full functionality' including locality scheduling, and homogeneous redundancy
  - Testing it with 'real' applications such as Chess, Game of Go, and etc.
- 

# Thanks! / Questions?

