

Distributed Chess AI Computation through RT-BOINC

- Sangho Yi, Emmanuel Jeannot, Derrick Kondo -

Related work #1: **Towards Real-Time, Many Task Applications on Large Distributed Systems** (Euro-Par 2010)

Related work #2: **RT-BOINC: A Massive Parallel Real-Time Computing Platform using Volunteered Resources** (will be submitted to FGCS)

Presenter: Sangho Yi (sangho.yi@inria.fr)

A dark silhouette of a tree is positioned in the bottom right corner of the slide, set against a dark background with a subtle gradient.

Introduction to RT-BOINC



RT-BOINC

is

a

Real-Time

BOINC.



RT-BOINC in a Nutshell

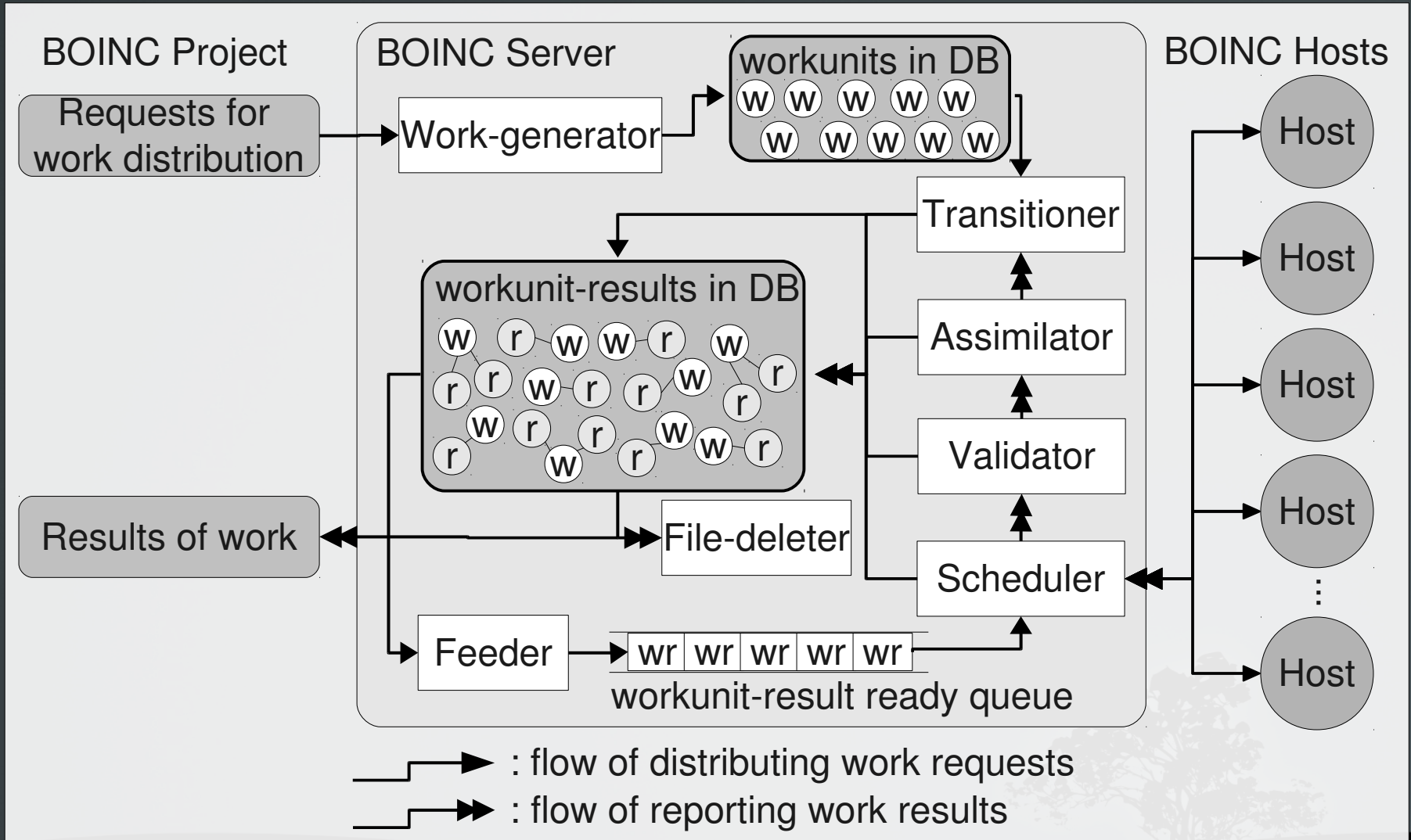
- RT-BOINC features
 - Low WCET (worst-case execution time) for all server-side operations (300~1,000 times faster)
 - No database operations at run-time
 - $O(1)$ interfaces for data structures
 - Reduced complexity for server daemons
 - Almost $O(1)$ (up to $\sim O(n)$)



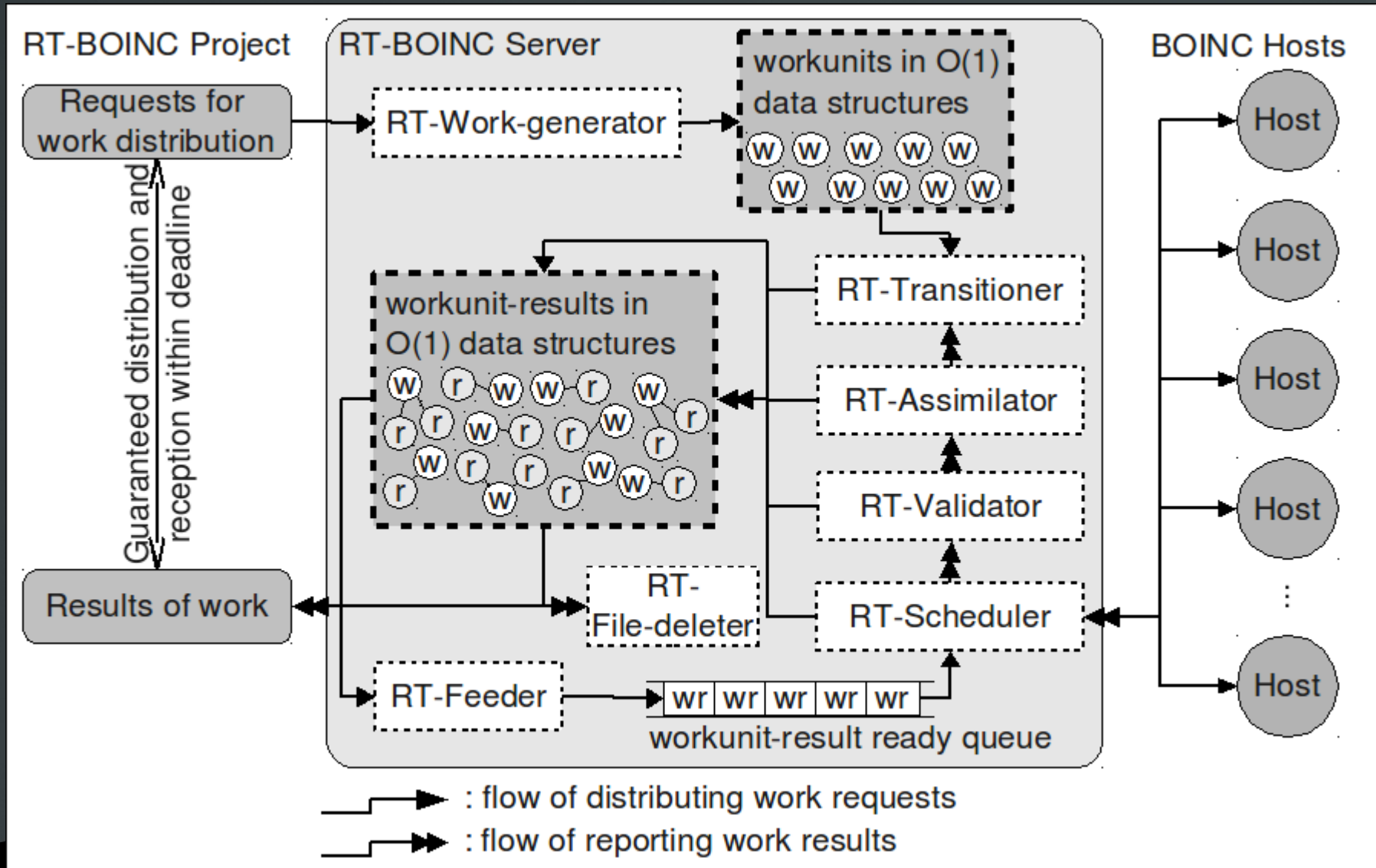
BOINC, RT-BOINC,
what's the difference?



Original BOINC Internal



RT-BOINC Internal



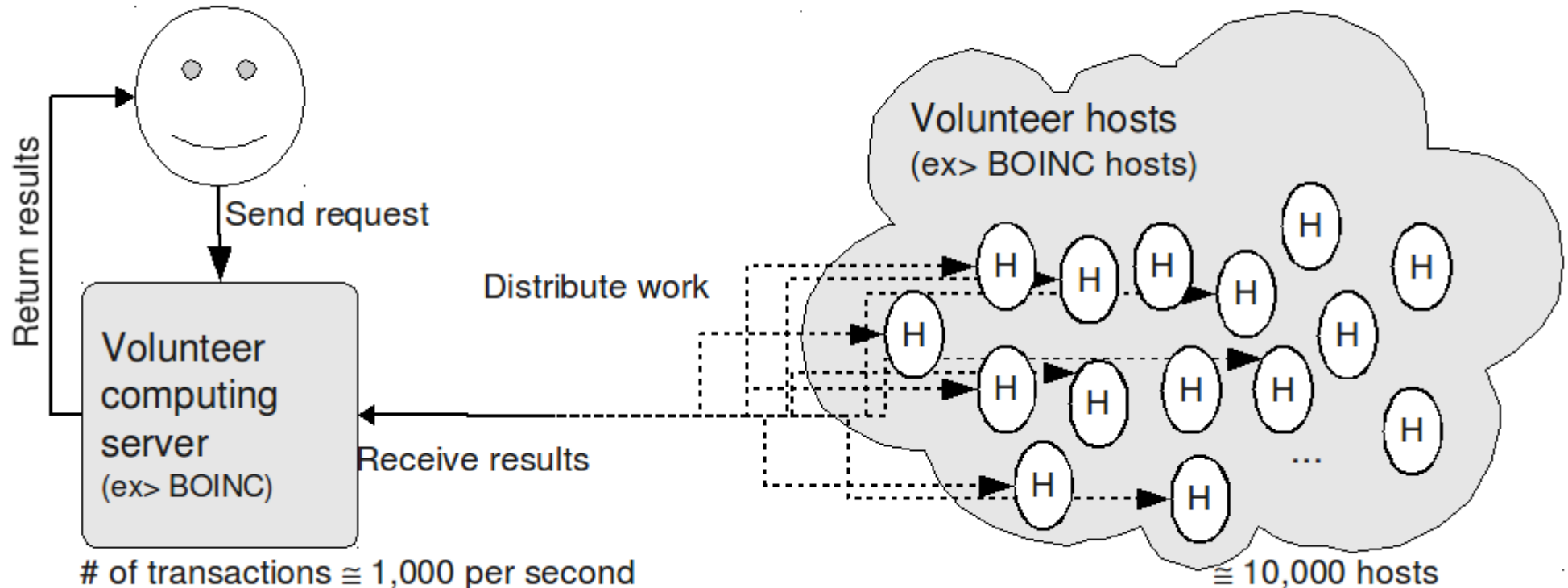
When do we need a RT-BOINC?



Short-term, real-time applications

- An example: "10-second-completion." - in a "Chess game"


Chess player's mission: Get next move within 10 seconds



Why cannot BOINC work?




Actually, BOINC ...

- BOINC is tailored for maximizing throughput, not minimizing latency.
 - BOINC does not support real-time task management in order of seconds.
 - BOINC supports only the order of days, or hours of deadline management.
 - BOINC normally handles <100 transactions per second
- 

But, BOINC can be tuned to
support short-term, real-time tasks,
isn't it?




Maybe Yes, Maybe No.

- BOINC has at least $O(\log n) \sim O(n)$ complexity for handling internal data records. (up to polynormials)
 - BOINC may work with small number of volunteers.
 - BOINC may **NOT** work more than a few hundreds cores of volunteered resources.
- 

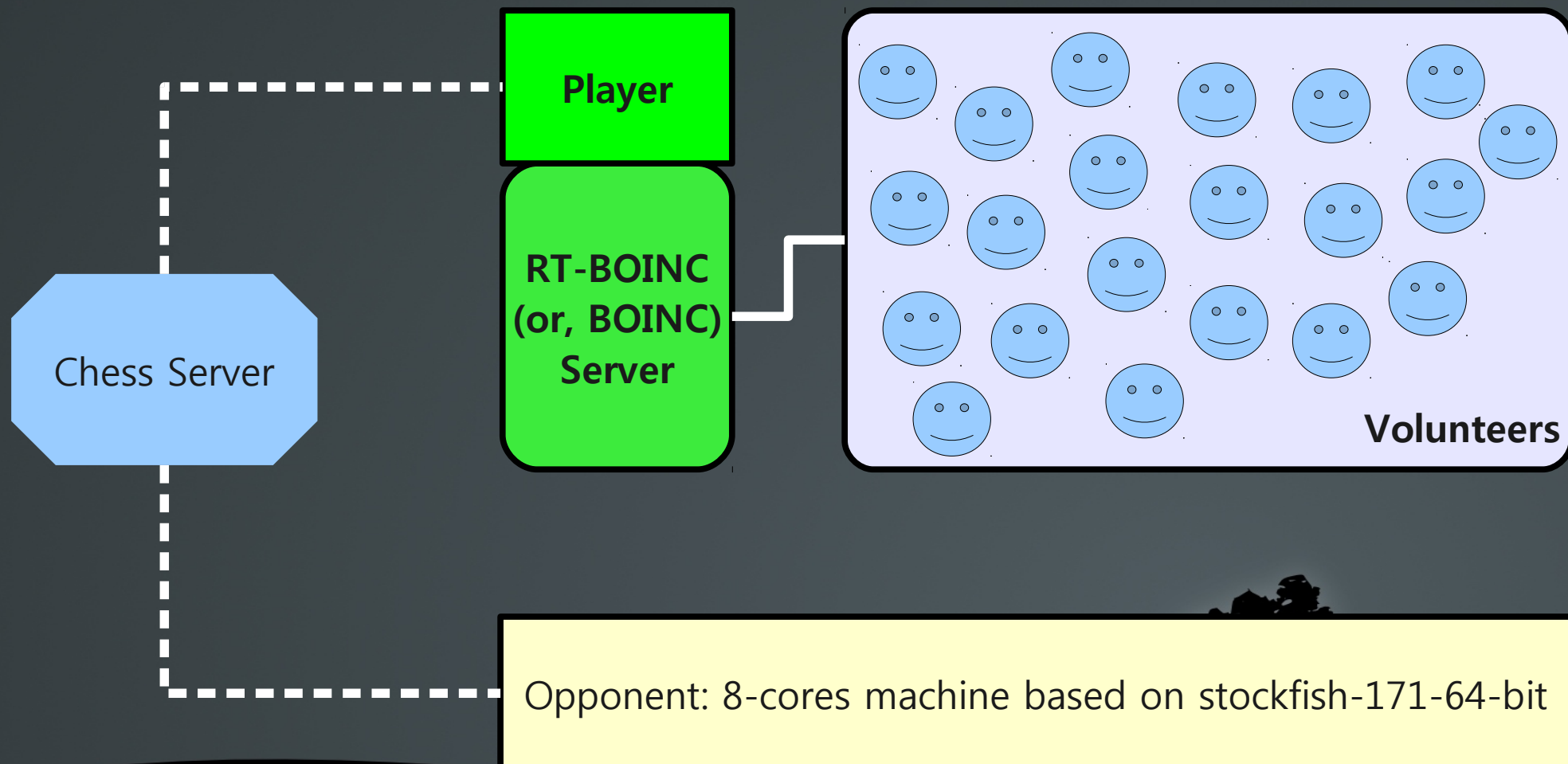
OK, show me the evidence!



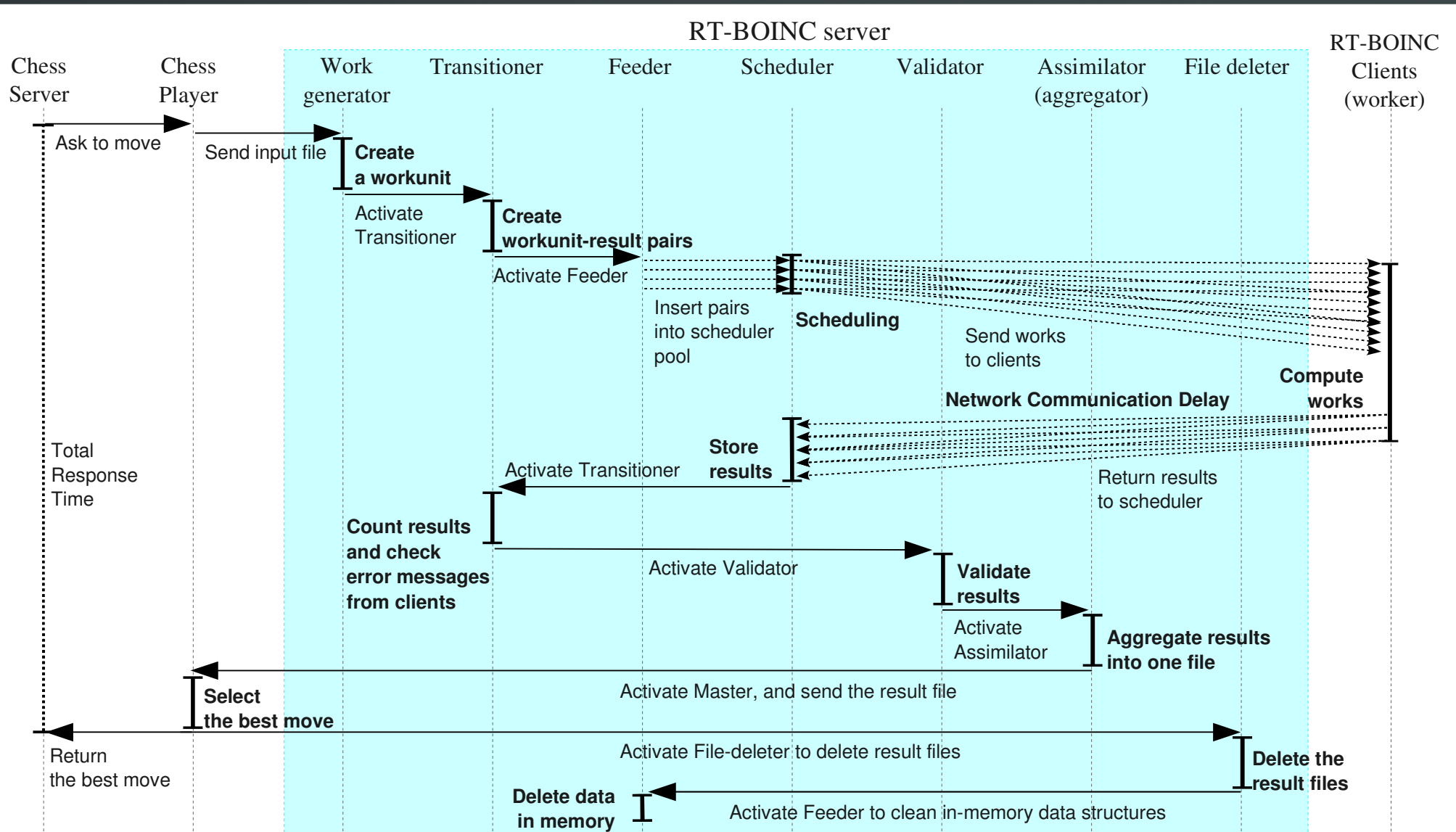
Case Study: Chess AI on RT-BOINC

- Chess AI:
 - Emmanuel Jeannot's Distributed Chess AI (based on stockfish-171-64-bit version)
 - Evaluation Method:
 - RT-BOINC (or, BOINC) plays Chess against one machine.
 - Evaluation Criteria:
 - Time for each move (BOINC, RT-BOINC)
 - Winning ratio (BOINC, RT-BOINC)
- 

How does it work?



How does it work in the server?



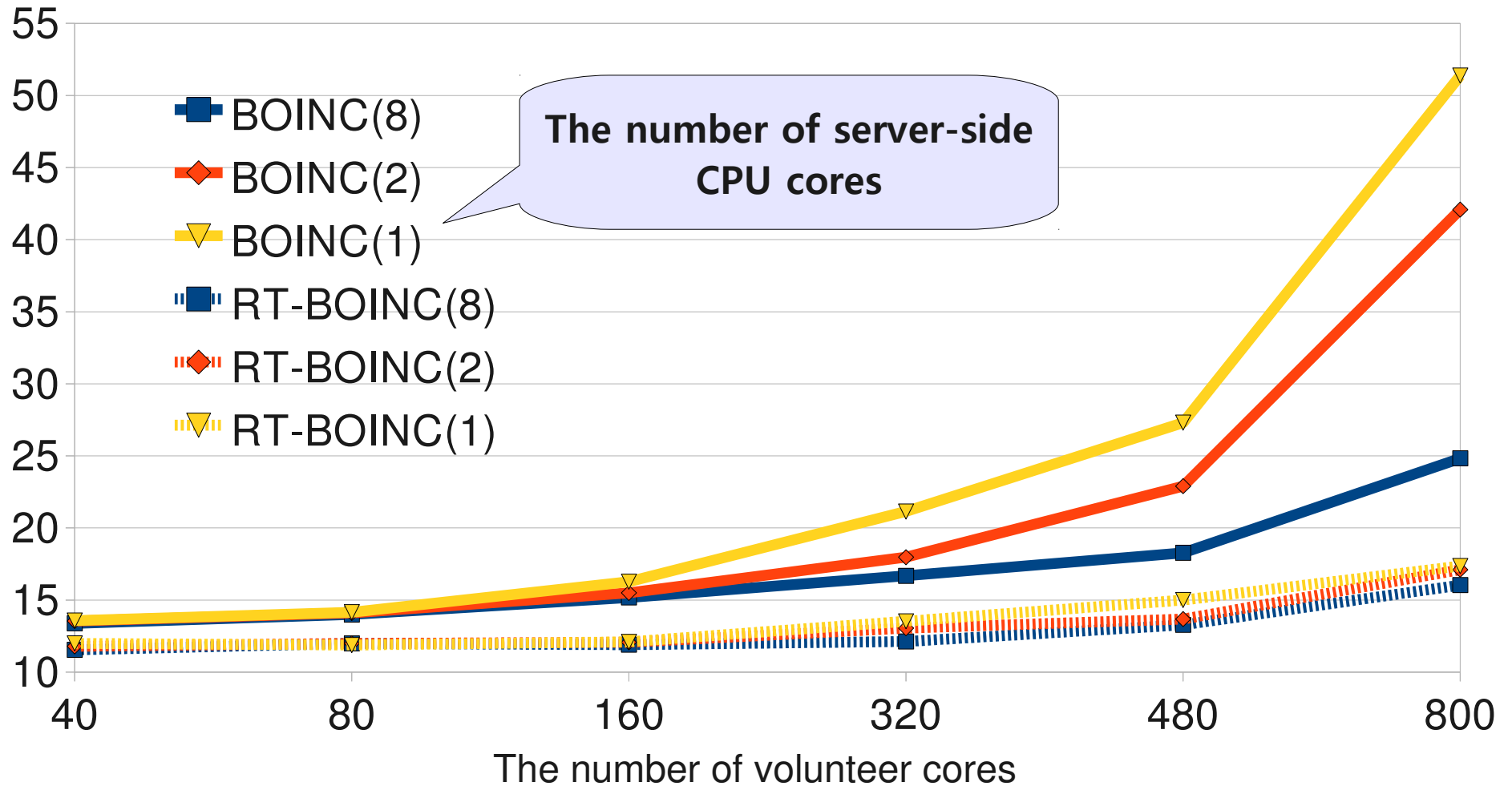
Parameters

- Number of volunteer CPU cores
 - 40, 80, 160, 320, 480, 800 cores (from G5K)
- Number of server-side CPU cores
 - 1, 2, 8 cores
- The amount of computation for each client
 - Fixed to 5 seconds

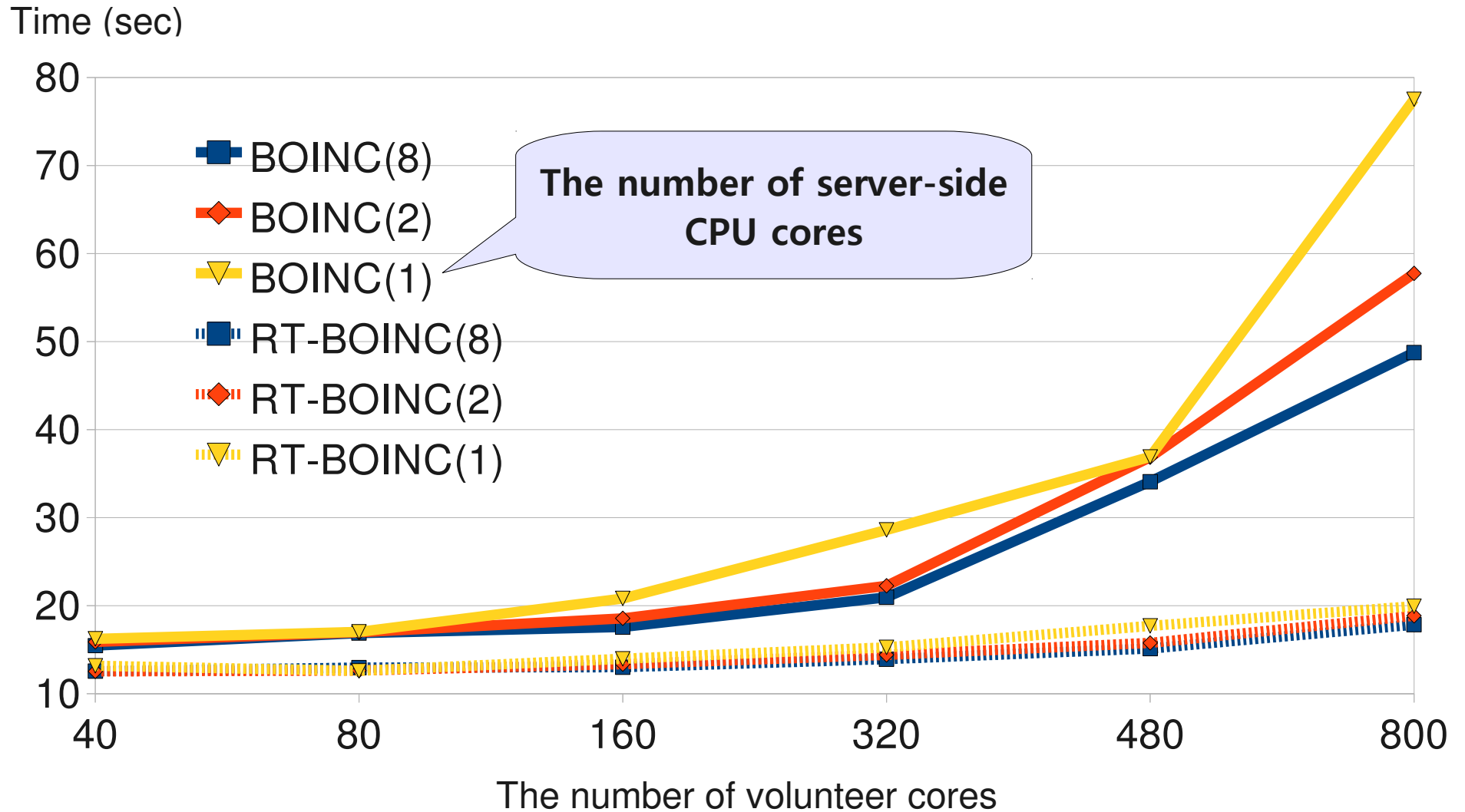


Average Response Time

Time (sec)

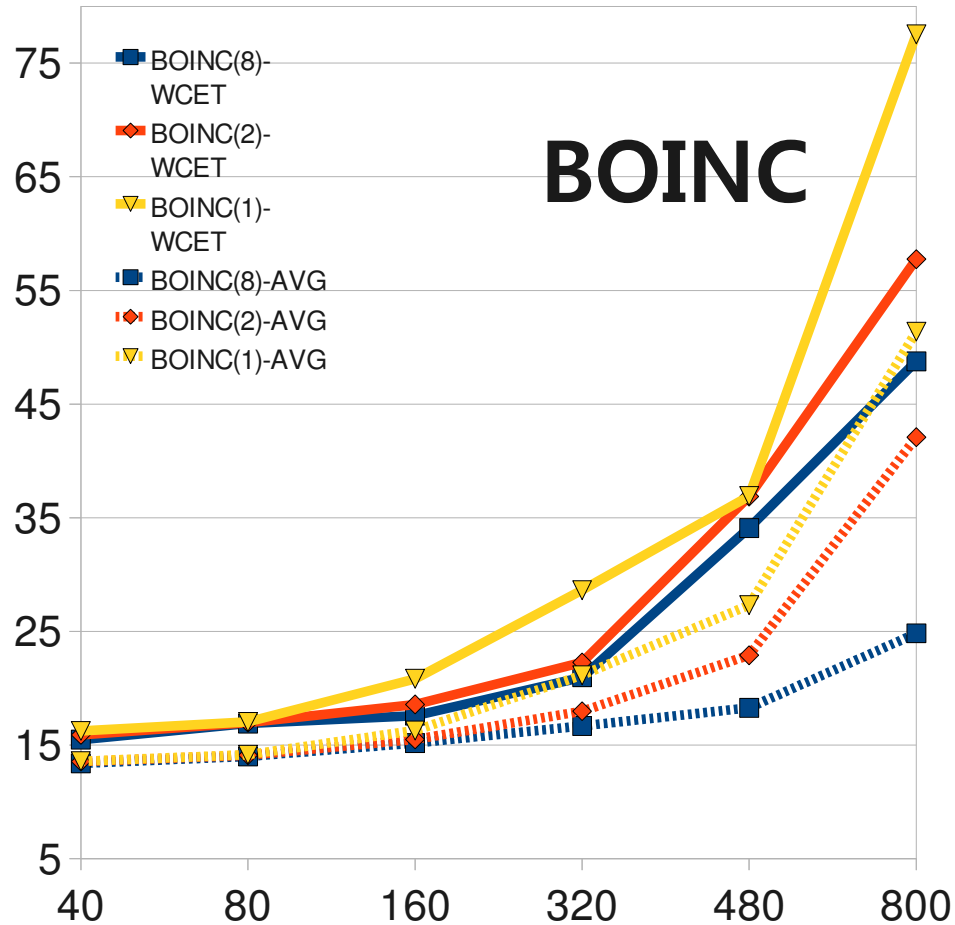


Worst-case Response Time

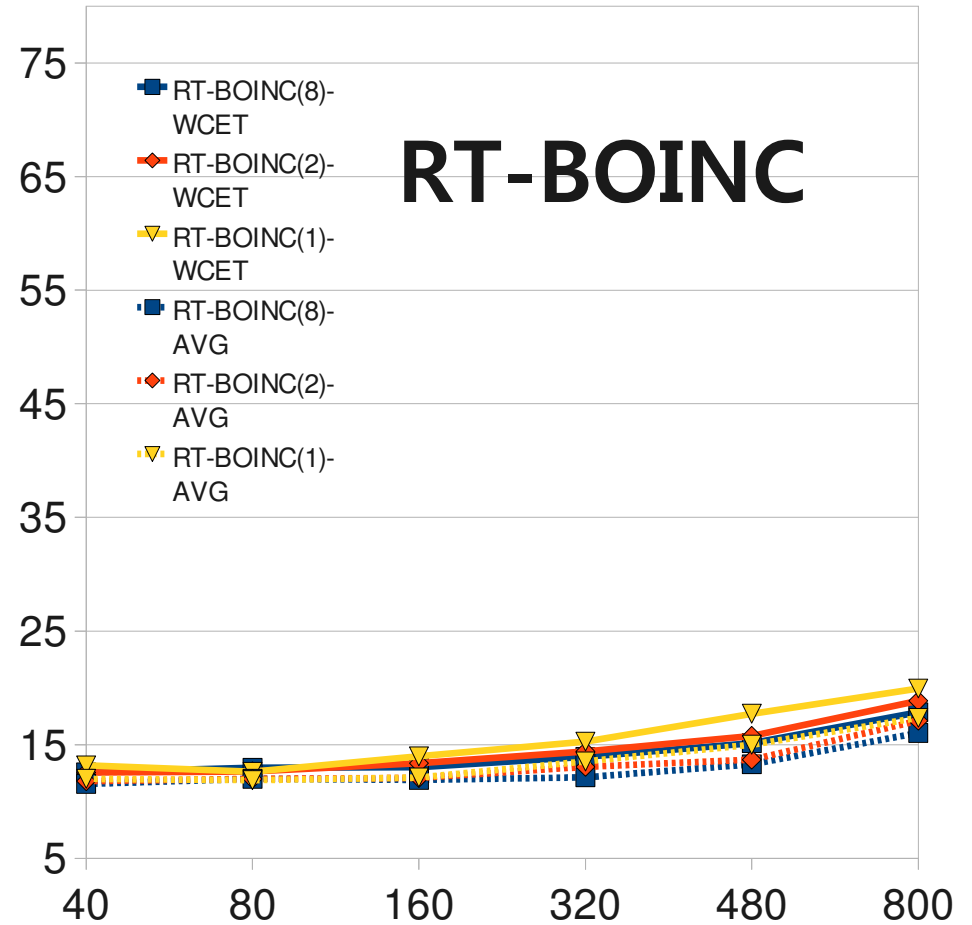


Performance Gap between Average and the Worst-case

Time (sec)



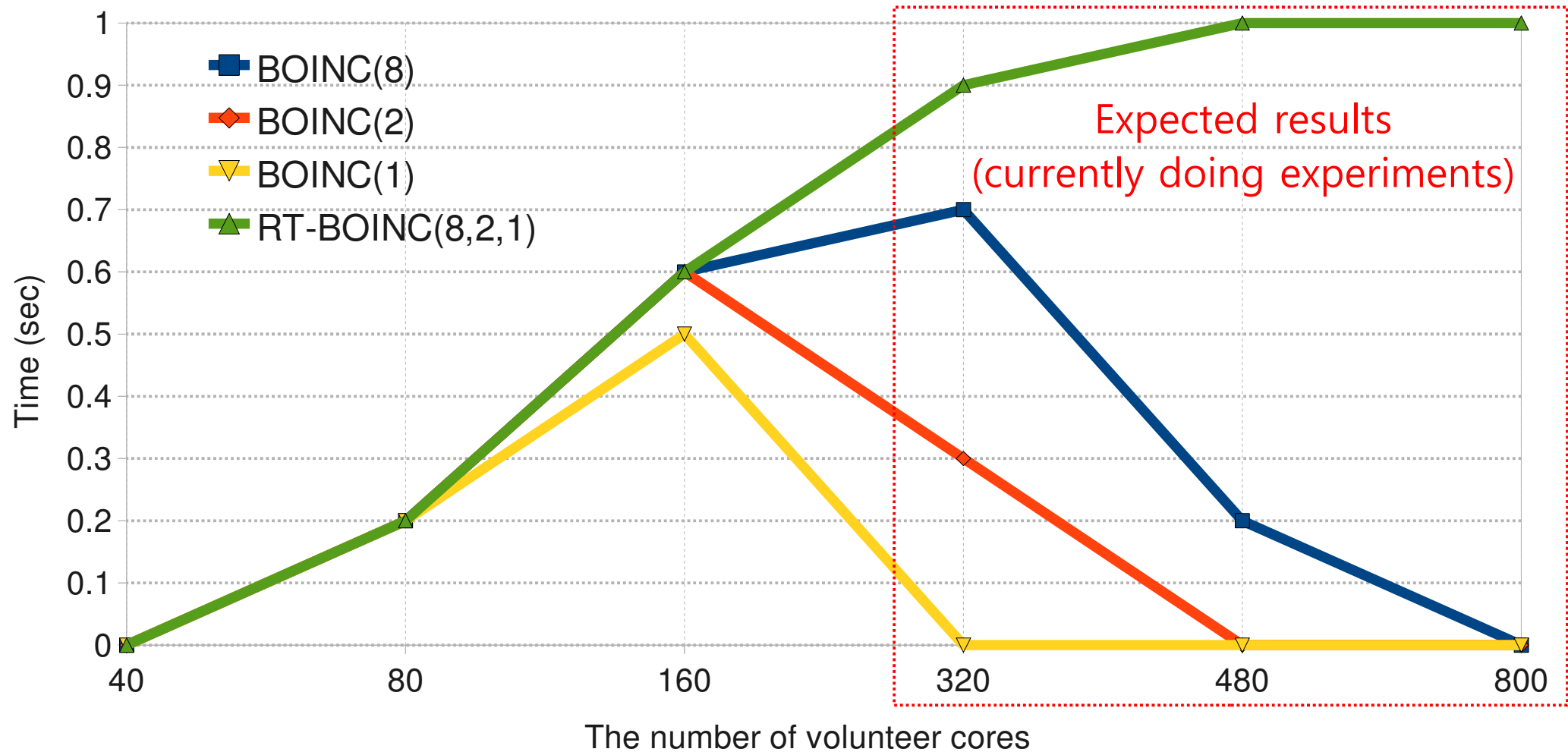
Time (sec)



The number of volunteer cores

Winning Ratio vs. Opponent

(where time constraint = 20s/move)



Can I get the source code of
RT-BOINC?



Documents 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 r), 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 icon, a text box for an optional review, and buttons for "Update Review" and "Delete Review".

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 r)
- 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

Does RT-BOINC have some remaining issues for future work?



Remaining Issues (1)

- Trade-off between Space and Time
 - More memory usage → Lower WCET
 - Less memory usage → Higher WCET
- Memory Management
 - Current RT-BOINC manages data records on a (static) shared memory segment.
 - Static management = space-inefficient, simple, faster
 - Dynamic management = space-efficient, complicated, slower when reallocating shared memory segment

Remaining Issues (2)

- Resource-aware Scheduling Policies
 - Latency-aware Scheduling → Lower distribution of network communication delay
 - Availability-aware Scheduling → Lower # of deadline misses in the presence of unavailability
- Better $O(1)$ data structure
 - To reduce memory space usage
 - To reduce WCET



Thank you!

(to be continued in the future...)

