A Prediction-based Approach to Distributed Interactive Applications

Peter A. Dinda

Jason Skicewicz Dong Lu

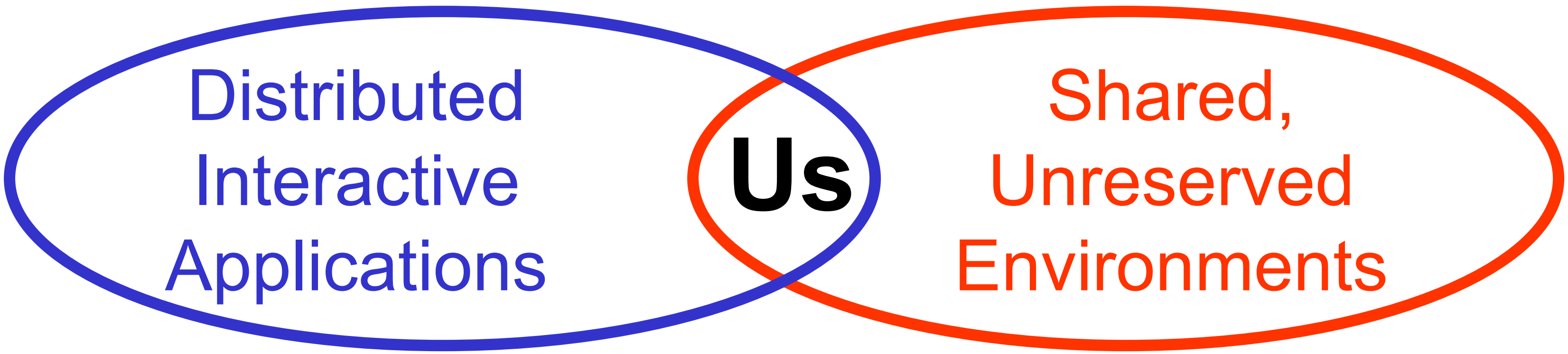
Prescience Lab

Department of Computer Science

Northwestern University

http://www.cs.northwestern.edu/~pdinda

Context and Question



How an **distributed interactive** application running on **shared, unreserved** computing environment provide **consistent responsiveness**?

Why Is This Interesting?

- Interactive resource demands set to explode
  - Tools and toys increasingly are physical simulations
  - High-performance computing for everyone
- People provision according to peak demand
  - Responsiveness tied to peak demand
  - 90% of the time CPU or network link is unused
- Opportunity to use the resources smarter
  - New kinds of applications
  - Shared resource pools, resource markets, Grid…

Interactivity Demands Responsiveness

But…

- Dynamically shared resources
  - Commodity environments
- Resource reservations unlikely
  - History
  - End-to-end requirements
- User-level operation
  - Difficult to change OS
  - Want to deploy anywhere

Supporting interactive apps under such constraints is not well understood

Approach

- Soft real-time model
  - Responsiveness requirement -> deadline
  - Advisory, no guarantees
- Adaptation mechanisms
  - Exploit DOF available in environment
- Prediction of resource supply and demand
  - Control the mechanisms to benefit the application
  - Computers as natural systems

Rigorous statistical and systems approach to prediction

Outline

- The story
- Interactive applications
  - Virtualized Audio
- Advisors and resource signals
- The RPS system
  - Intermixed discussion and performance results
- Current work
  - Wavelet-based techniques

All Software and Data publicly available

Application Characteristics

- Interactivity
  - Users initiate aperiodic tasks with deadlines
  - Timely, consistent, and predictable feedback needed before next task can be initiated
- Resilience
  - Missed deadlines are acceptable
- Distributability
  - Tasks can be initiated on any host
- Adaptability
  - Task computation and communication can be adjusted

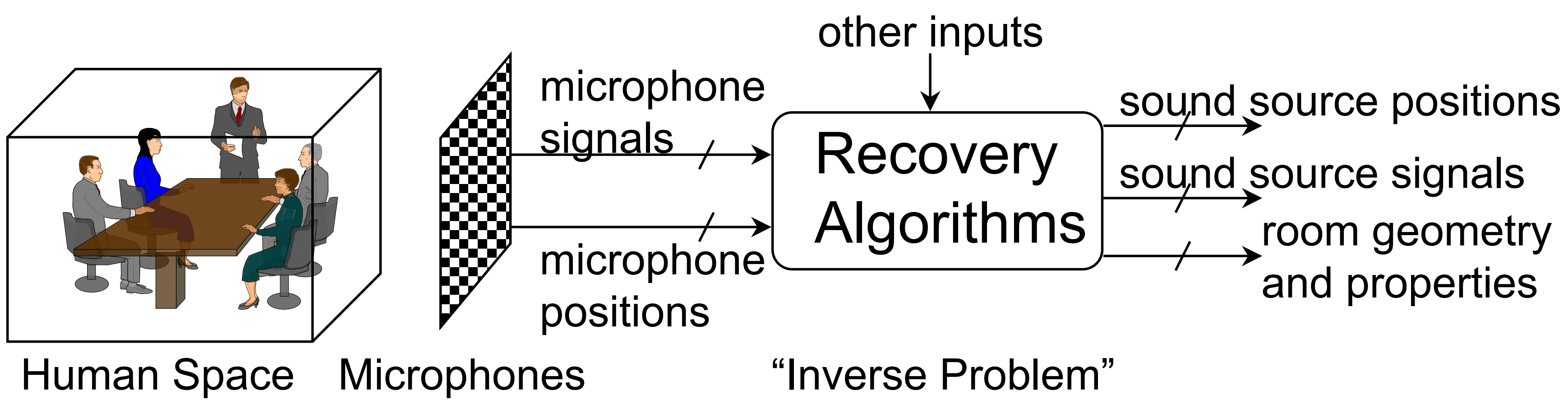
Shared, unreserved computing environments

Applications

- Virtualized Audio
    - Dong Lu
- Image Editing
- Games
- Visualization of massive datasets
  - Interactivity Environment at Northwestern
    - With Watson, Dennis
  - Dv project at CMU

VA: The Inverse Problem

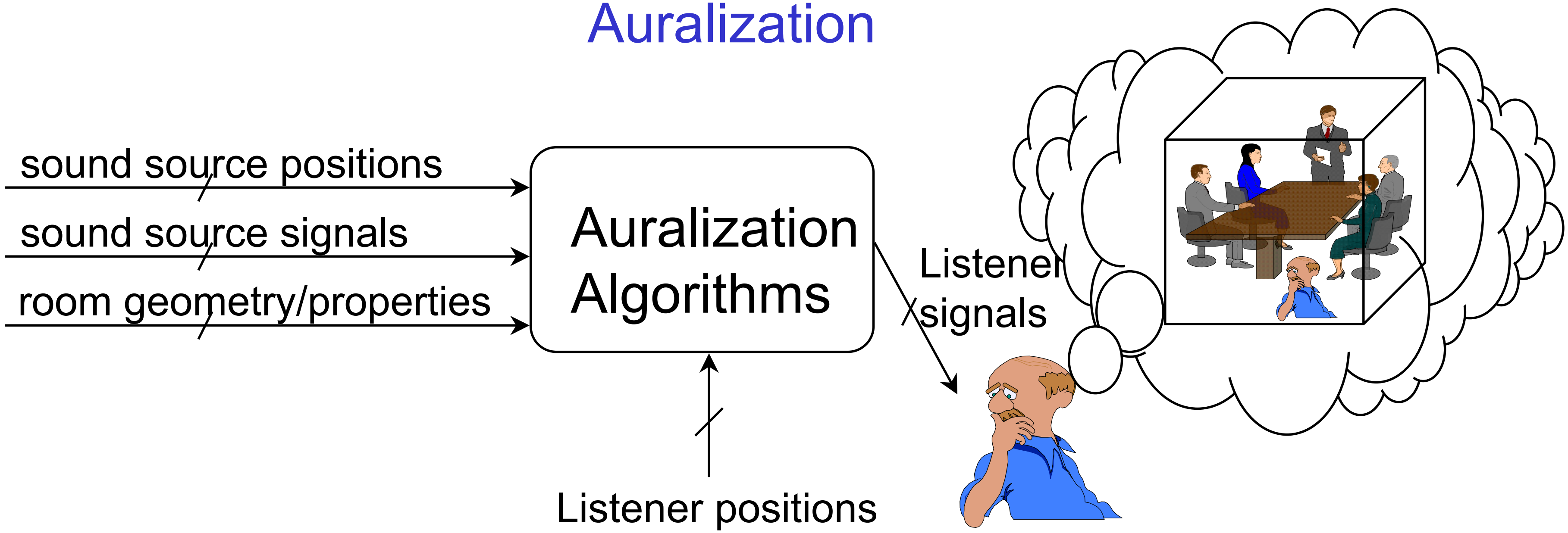
Source Separation and Deconvolution



- Microphone signals are a result of sound source signals, positions, microphone positions, and the geometry and material properties of the room.
- We seek to recover these underlying producers of the microphone signals.

VA: The Forward Problem

Auralization

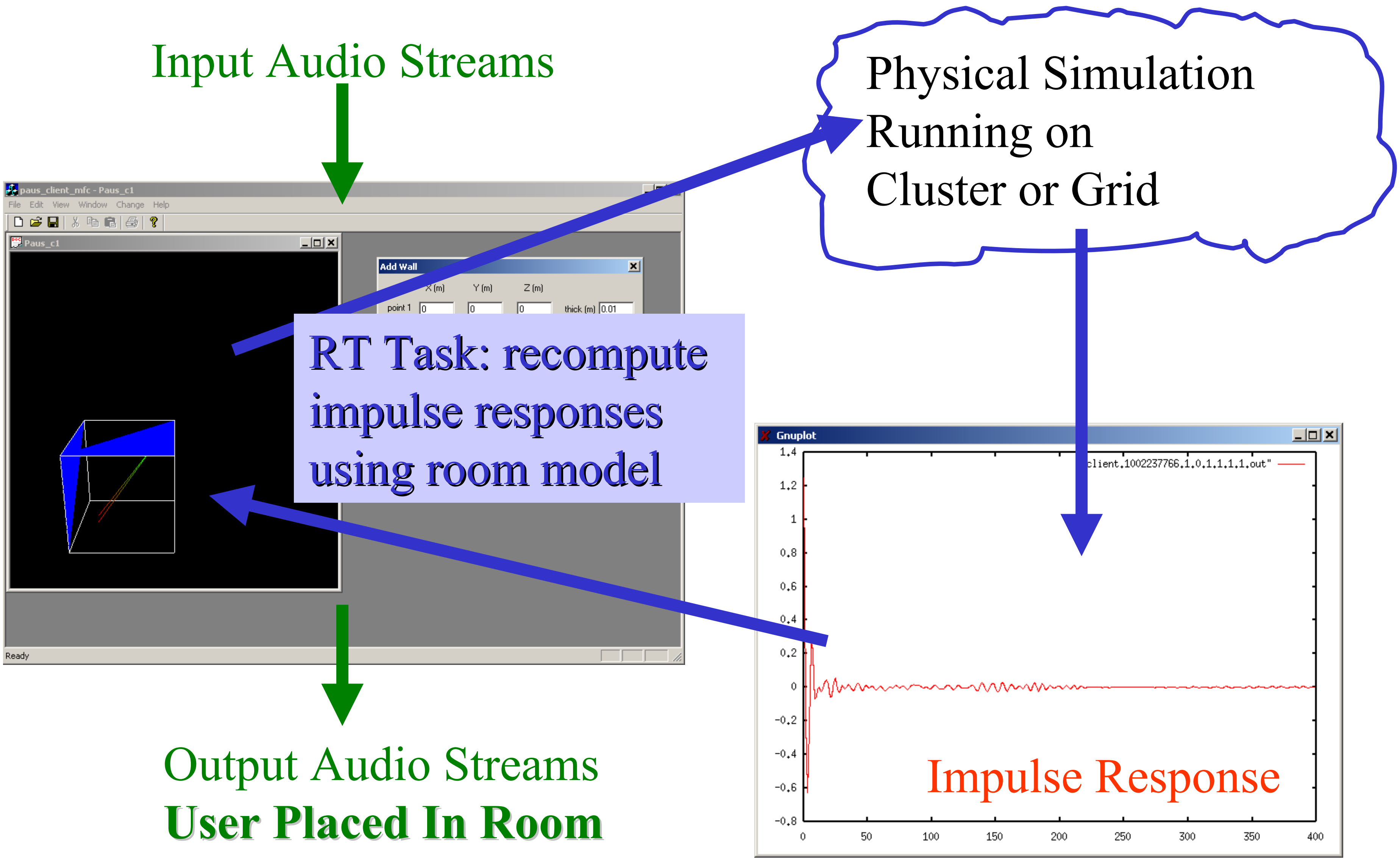


Listener wearing headphones

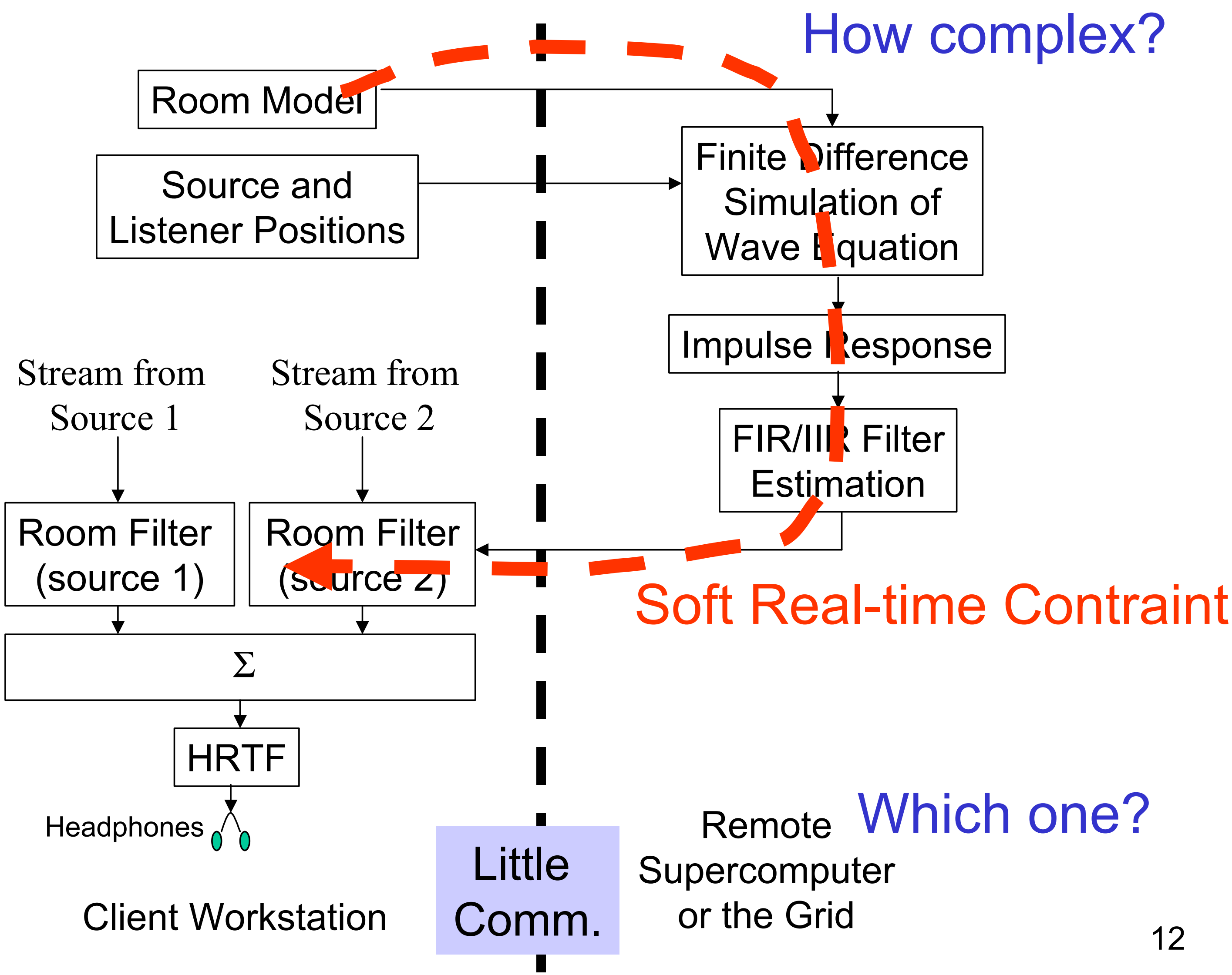
- In general, all inputs are a function of time
- Auralization must proceed in real-time

(AccessGrid 2001)

Forward Problem App Structure

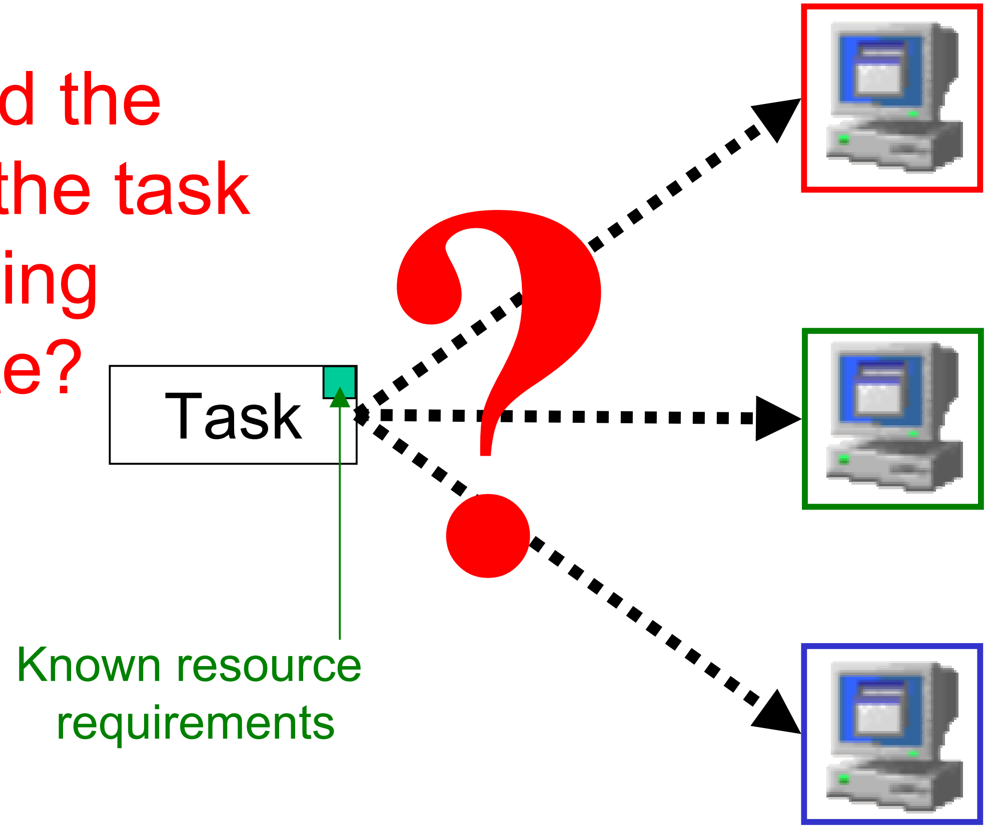


Forward Problem App Structure



A Universal Problem

Which host should the application send the task to so that its running time is appropriate?



What will the running time be if I...

Advisors

- Adaptation Advisors
  - Real-time Scheduling Advisor
    - Which host should I use?
    - Task assumptions appropriate to interactive applications
    - Soft real-time
    - Known resource demand
    - Best-effort semantics
- Application-level Performance Advisors
  - Running Time Advisor
    - What would running time of task on host x be?
    - Confidence intervals
    - Can build different adaptation advisors
  - Message Transfer Time Advisor ← Current focus
    - How long to transfer N bytes from A to B?

Resource Signals

- Characteristics
  - Easily measured, time-varying scalar quantities
  - Strongly correlated with resource supply
  - Periodically sampled (discrete-time signal)
- Examples
  - **Host load (Digital Unix 5 second load average)**
  - Network flow bandwidth and latency

Leverage existing statistical signal analysis and prediction techniques

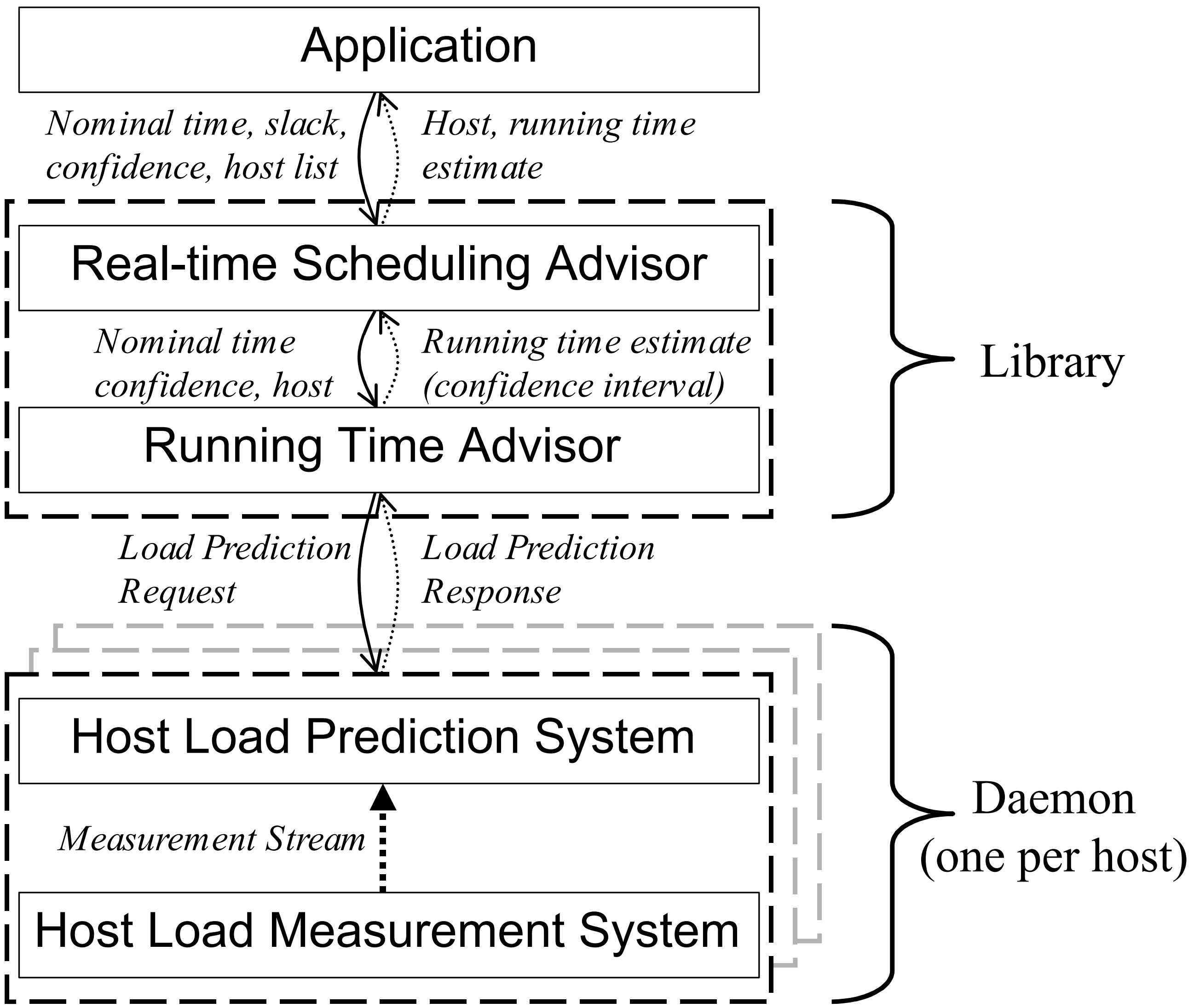
Currently: Linear Time Series Analysis and Wavelets

RPS Toolkit

- Extensible toolkit for implementing resource signal prediction systems [CMU-CS-99-138]
    - Growing: RTA, RTSA, Wavelets, GUI, etc
- Easy "buy-in" for users
    - C++ and sockets (no threads)
    - Prebuilt prediction components
    - Libraries (sensors, time series, communication)
- Users have bought in
    - Incorporated in CMU Remos, BBN QuO
    - A number of research users
- RELEASED

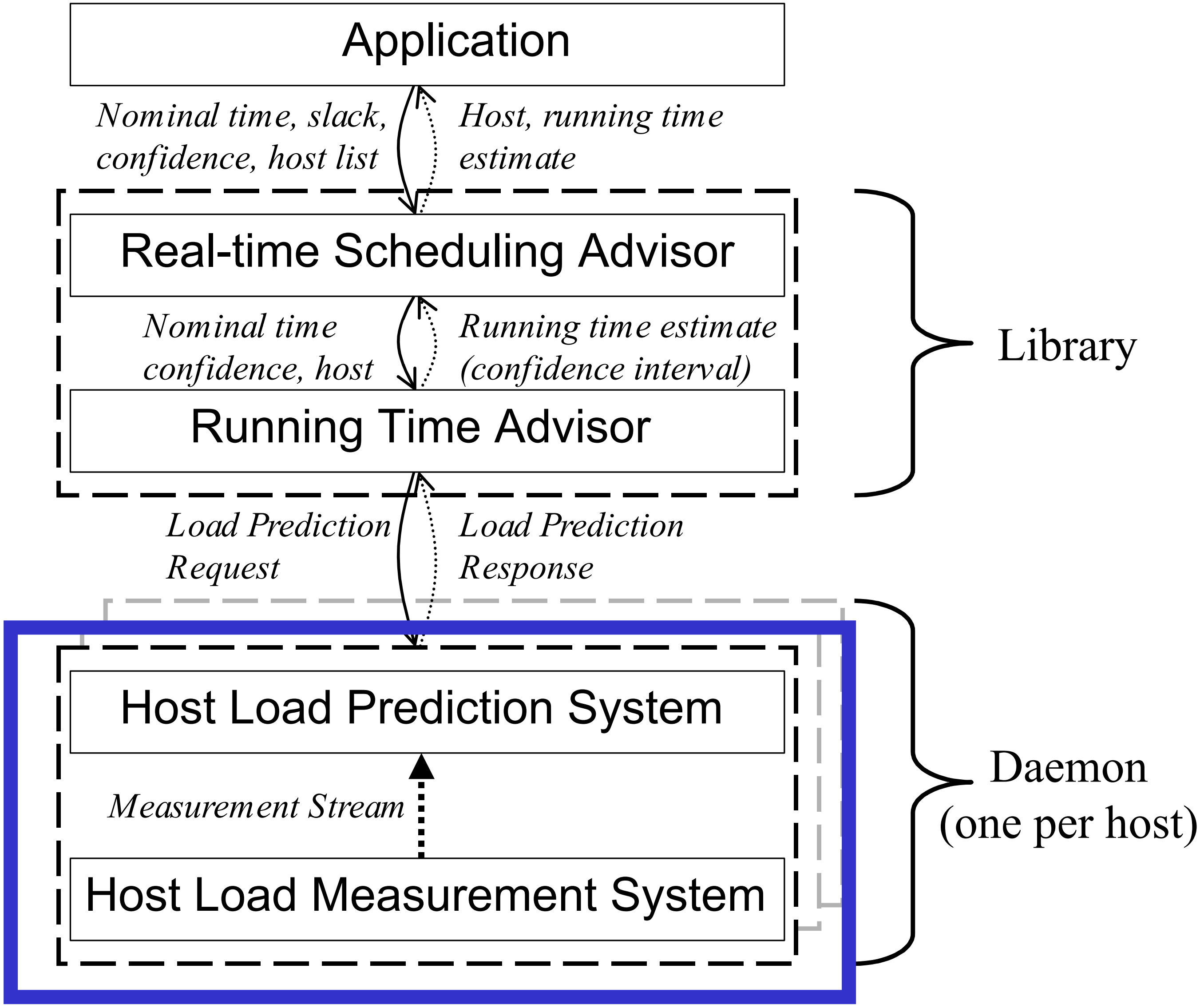
http://www.cs.northwestern.edu/~RPS

Example RPS System

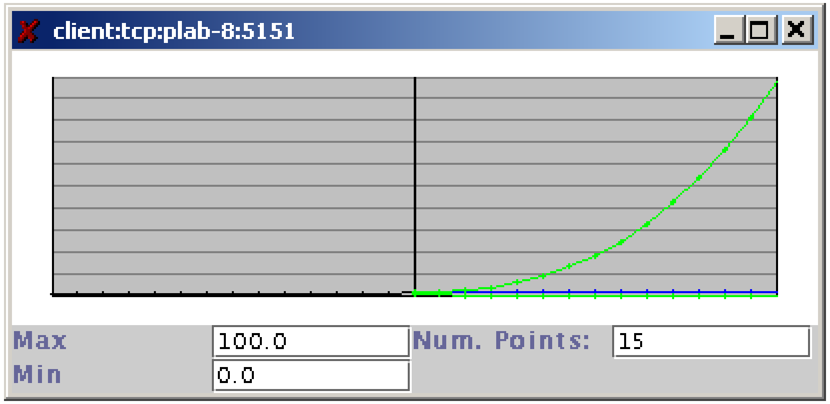


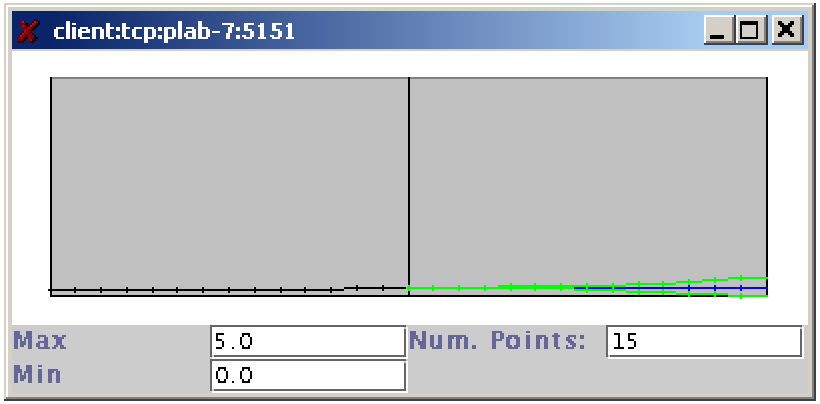
RPS components can be composed in other ways

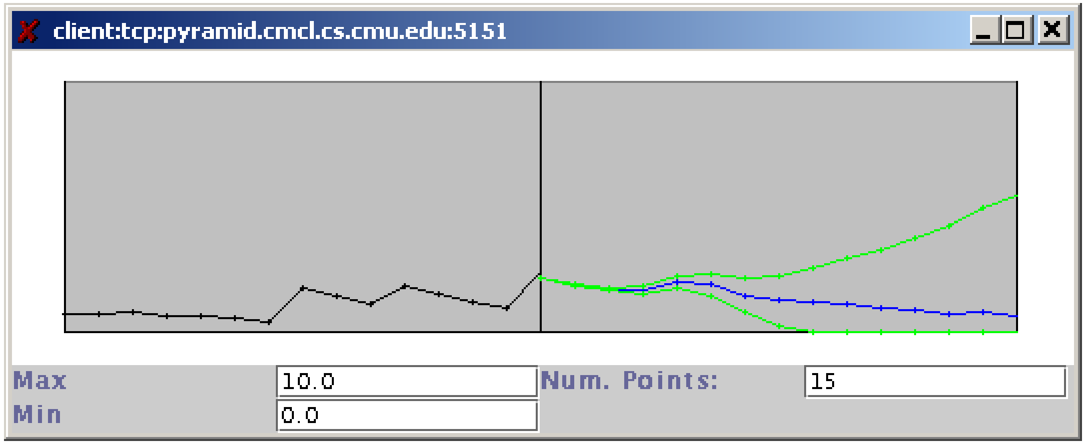
Example RPS System



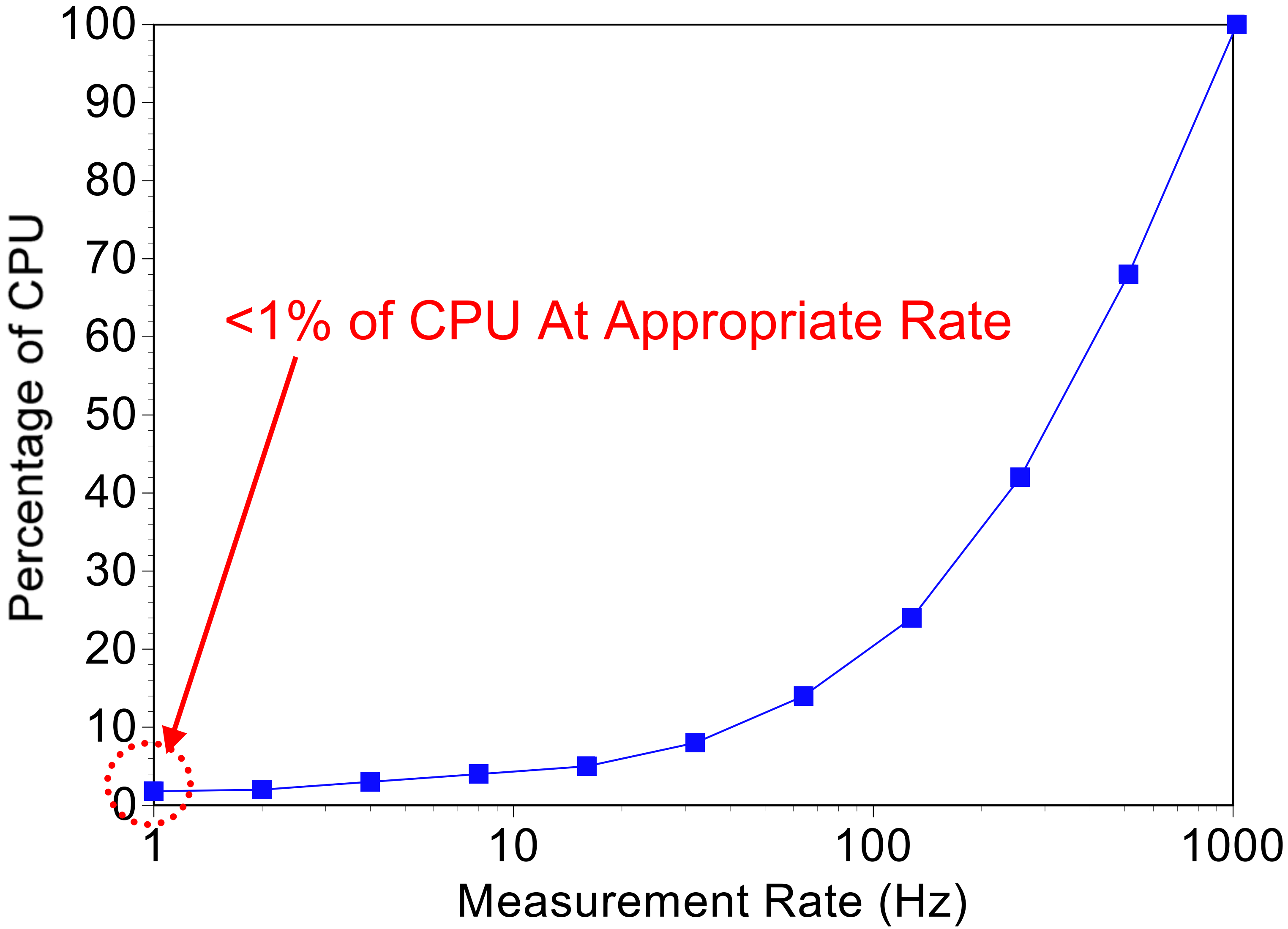
Measurement and Prediction







Measurement and Prediction Overhead



1-2 ms latency from measurement to prediction
2KB/sec transfer rate

Host Load Traces

- DEC Unix 5 second exponential average
  - 1 Hz
  - Playload tool

	Machines	Duration
August 1997	13 production cluster 8 research cluster 2 compute servers 15 desktops	~ one week (over one million samples)
March 1998	13 production cluster 8 research cluster 2 compute servers 11 desktops	~ one week (over one million samples)

http://www.cs.northwestern.edu/~pdinda/LoadTraces
http://www/cs.northwestern.edu/~pdinda/LoadTraces/playload

Salient Properties of Host Load

+/- Extreme variation

\+ Significant autocorrelation

Suggests appropriateness of linear models

\+ Significant average mutual information

\- Self-similarity / long range dependence

+/- Epochal behavior

- \+ Stable spectrum during an epoch
- \- Abrupt transitions between epochs

\+ encouraging for prediction

\- discouraging for prediction

(Detailed study in LCR98, SciProg99)

Linear Time Series Models

Model Class	Fit time (ms)	Step time (ms)	Notes
MEAN	0.03	0.003	Error is signal variance
LAST	0.75	0.001	Last value is prediction
BM(p)	46.26	0.001	Average over best window
AR(p)	4.20	0.149	Deterministic algorithm
MA(q)	6501.72	0.015	Function Optimization
ARMA(p,q)	77046.22	0.034	Function Optimization
ARIMA(p,d,q)	53016.77	0.045	Non-stationarity, FO
ARFIMA(p,d,q)	3692.63	9.485	Long range dependence, MLE

Pole-zero / state-space models capture autocorrelation parsimoniously

(2000 sample fits, largest models in study, 30 secs ahead)

AR(p) Models

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \ldots + \phi_p z_{t-p} + a_t$$

next value

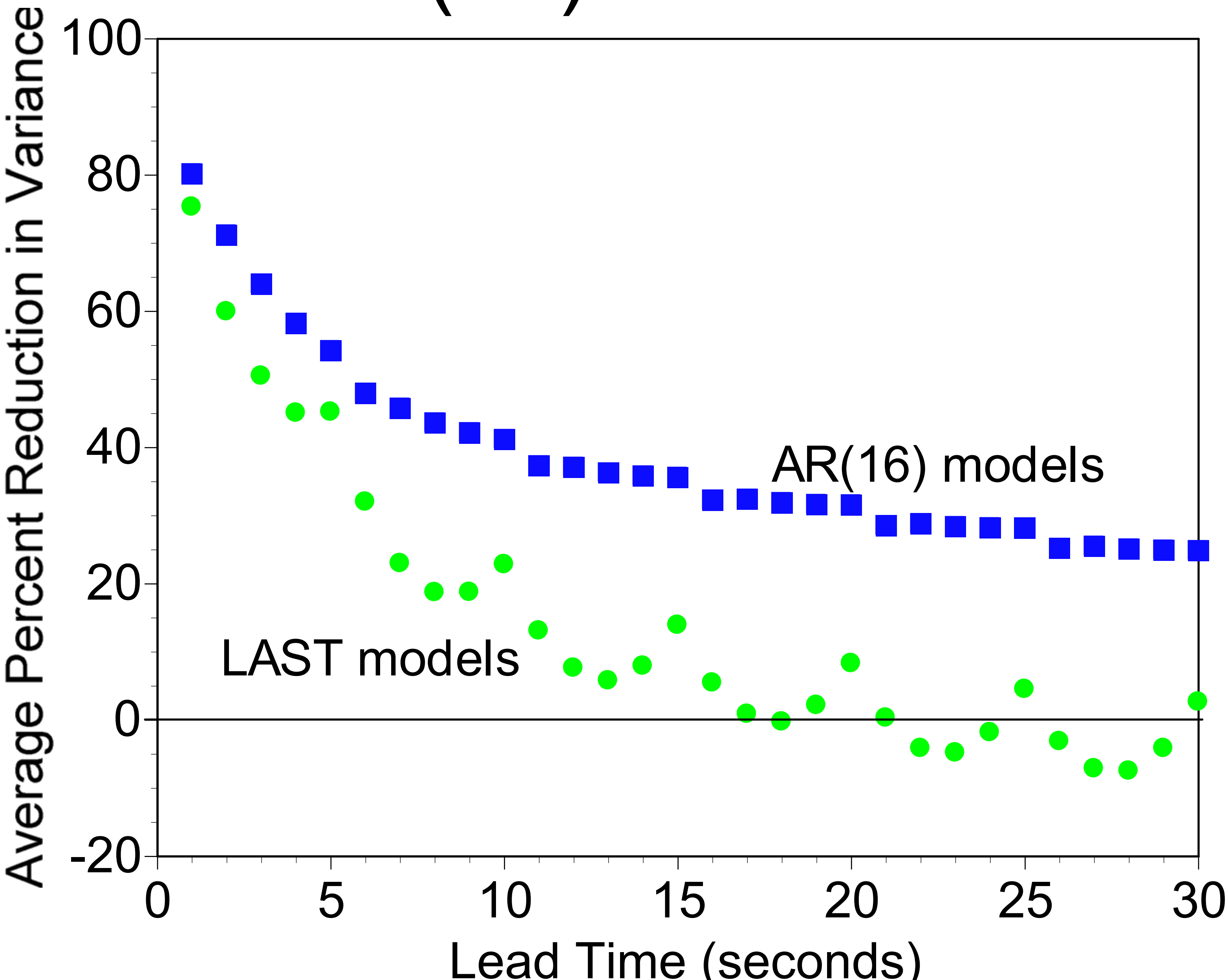
weights chosen to minimize mean square error for fit interval

p previous values

error

– Fast to fit (4.2 ms, AR(32), 2000 points)
– Fast to use (<0.15 ms, AR(32), 30 steps ahead)
– Potentially less parsimonious than other models

AR(16) vs. LAST



Average Percent Reduction in Variance
100
80
60
40
20
0
-20
0
5
10
15
20
25
30
Lead Time (seconds)
AR(16) models
LAST models

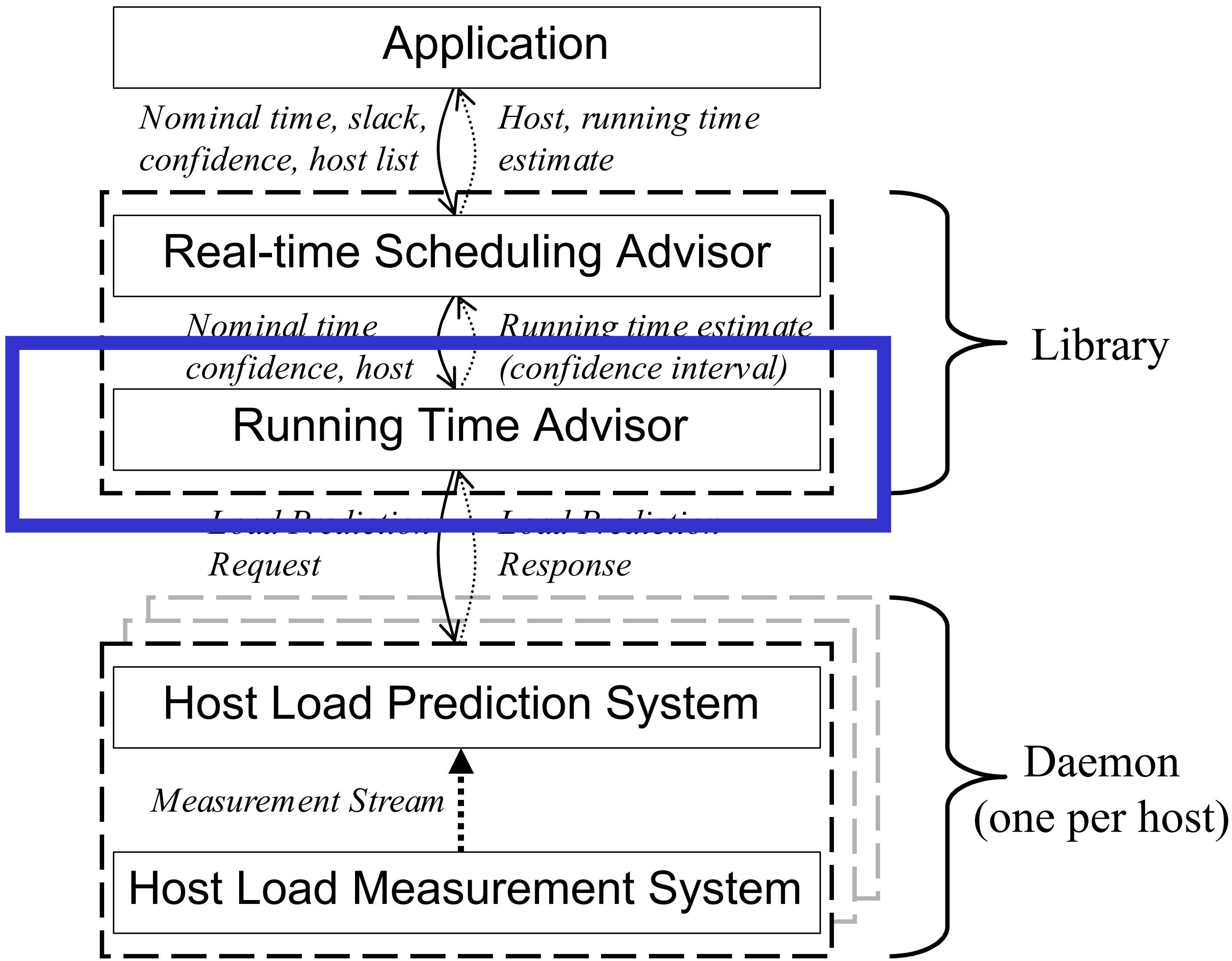
Host Load Prediction Results

- Host load exhibits complex behavior
  - Strong autocorrelation, self-similarity, epochal behavior
- Host load is predictable
  - 1 to 30 second timeframe
- Simple linear models are sufficient
  - Recommend AR(16) or better
- Low overhead

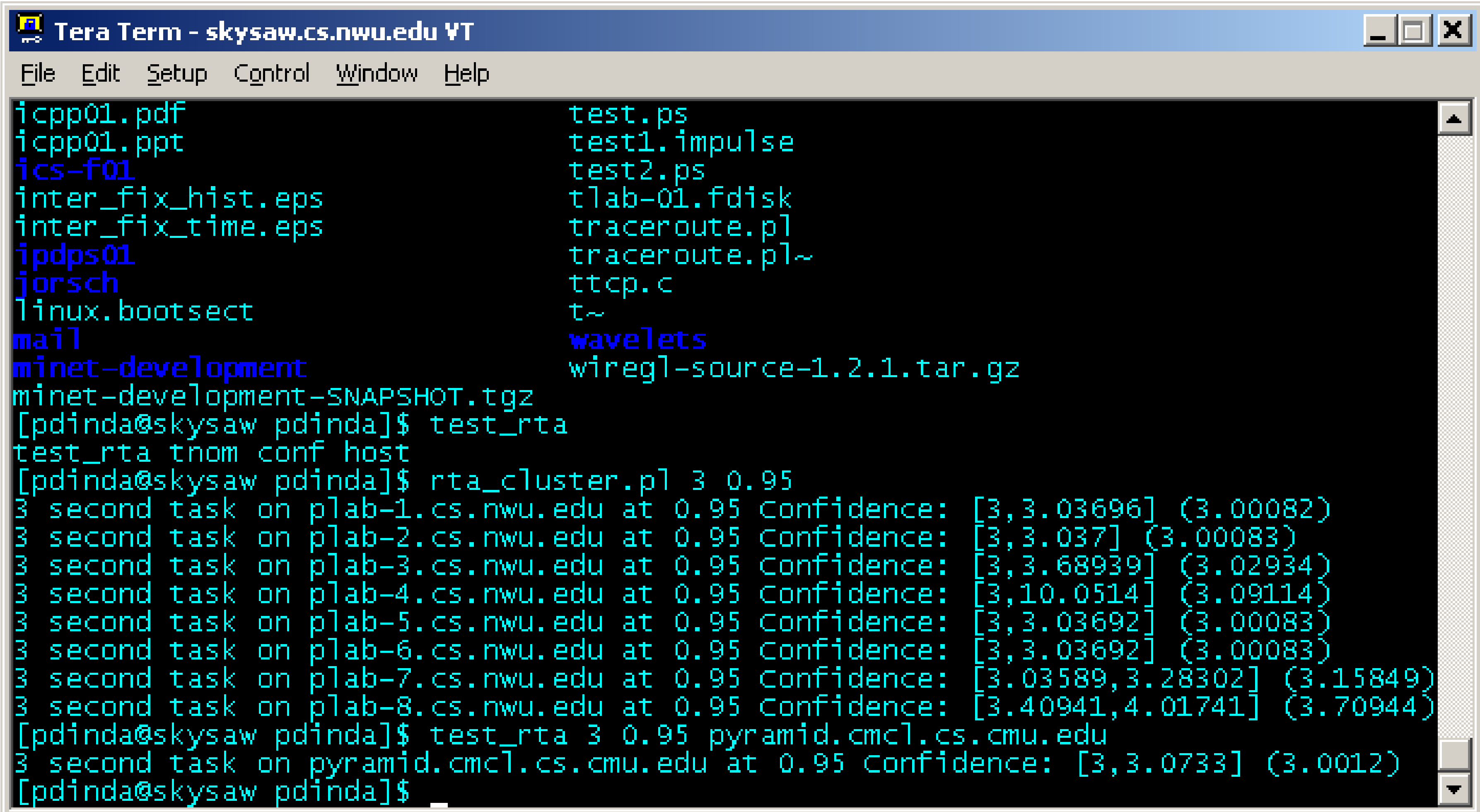
Extensive statistically rigorous randomized study

(Detailed study in HPDC99, Cluster Computing 2000)

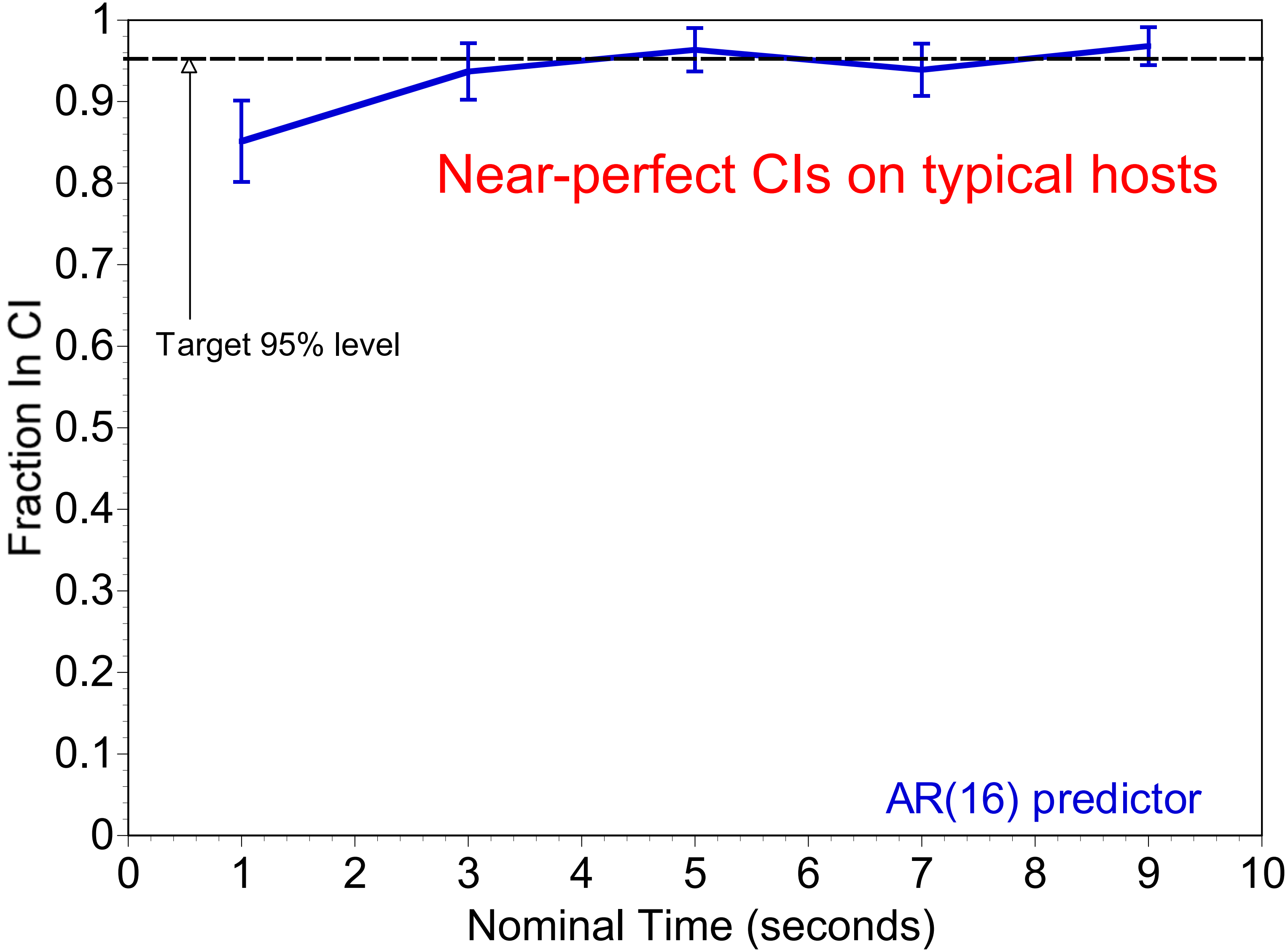
Example RPS System



Running Time Advisor



Example Performance



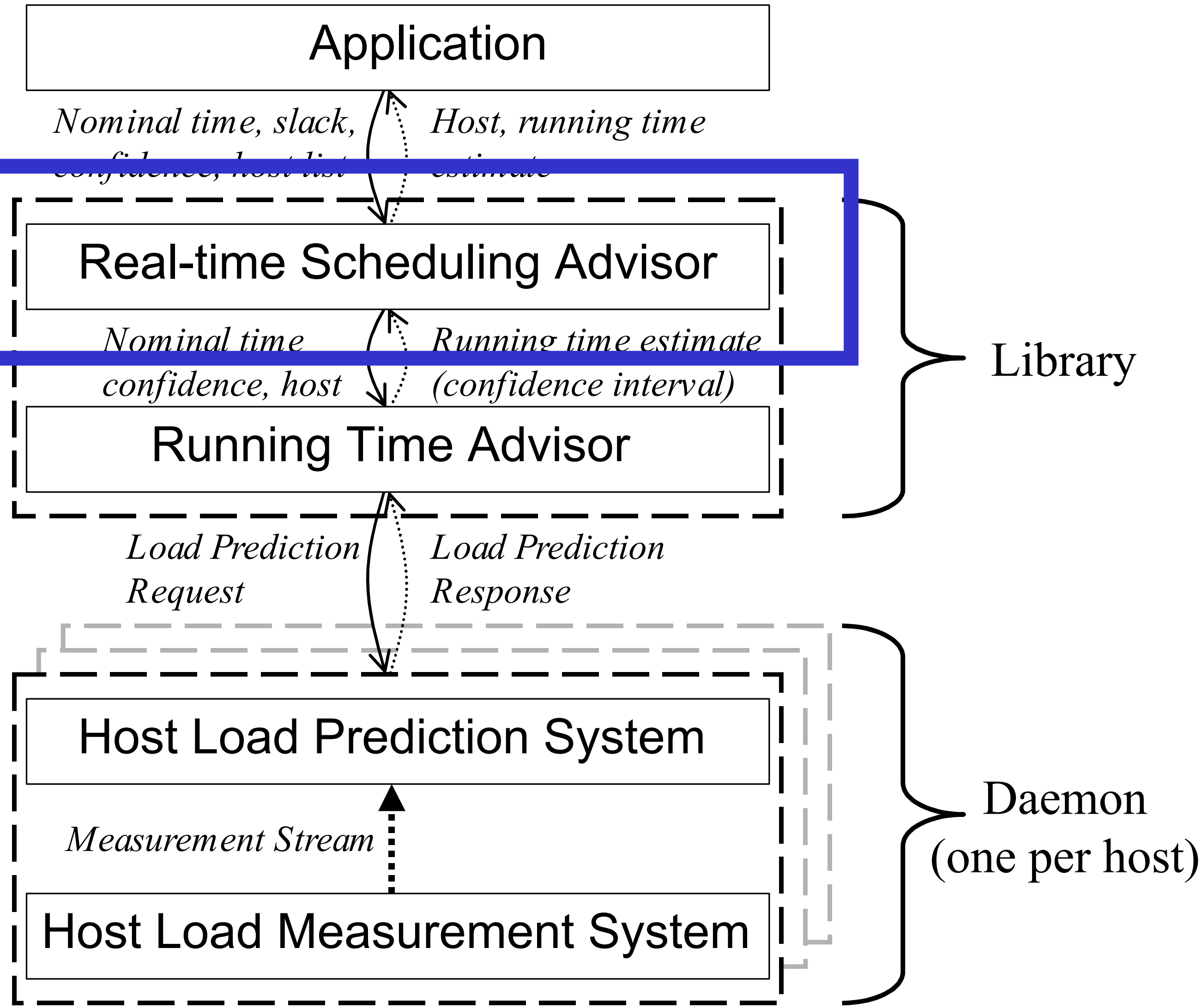
3000 randomized tasks

Running Time Advisor Results

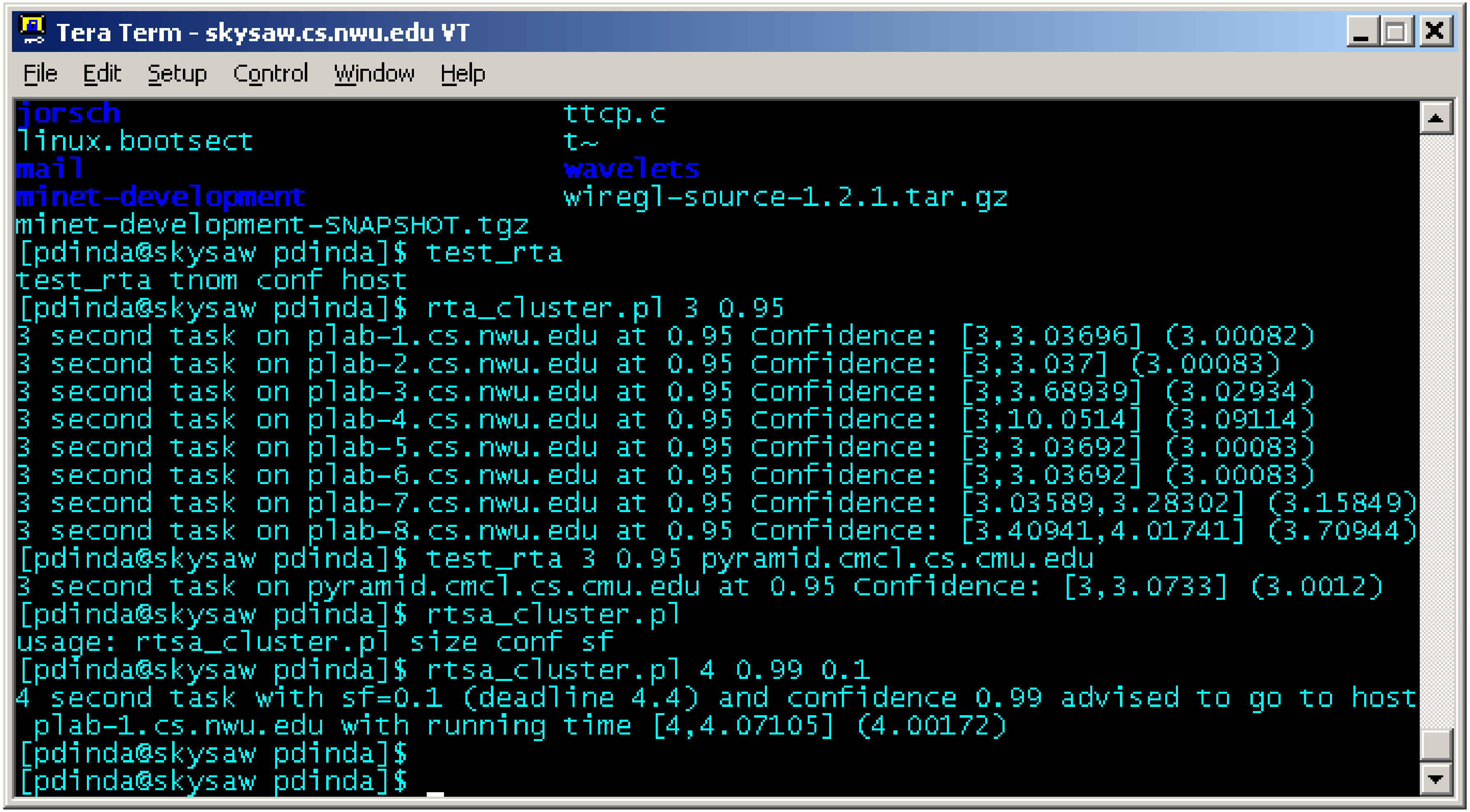
- Predict running time of task
  - Application supplies task size and confidence level
  - Task is compute-bound *(current limit)*
- Prediction is a confidence interval
  - Expresses prediction error
  - Statistically valid decision-making
- Maps host load predictions and task size through simple model of scheduler
  - Rigorous underlying prediction system essential
- Effective
  - Statistically rigorous randomized evaluation

(Study in HPDC 2001, SIGMETRICS 2001)

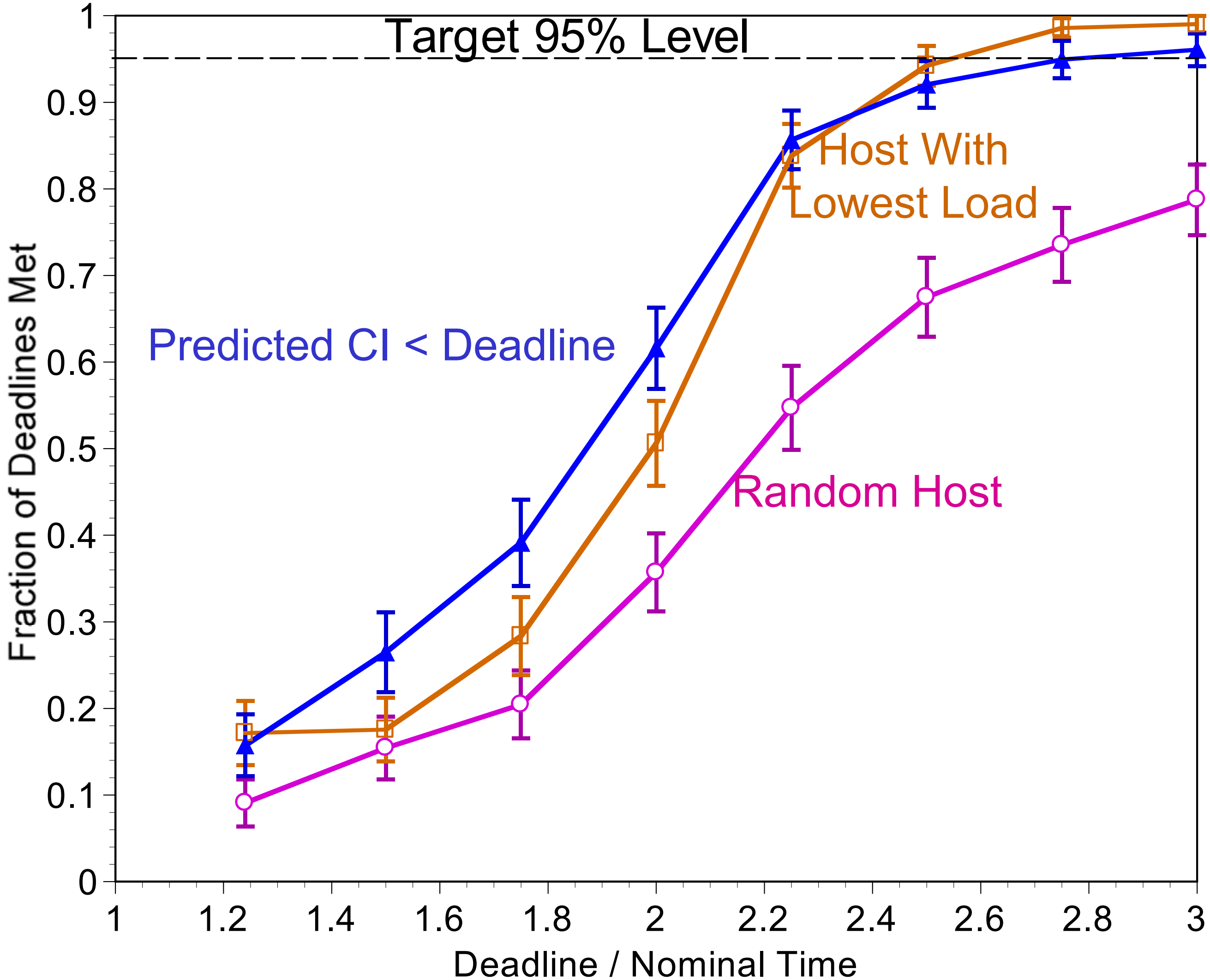
Example RPS System



Real-time Scheduling Advisor

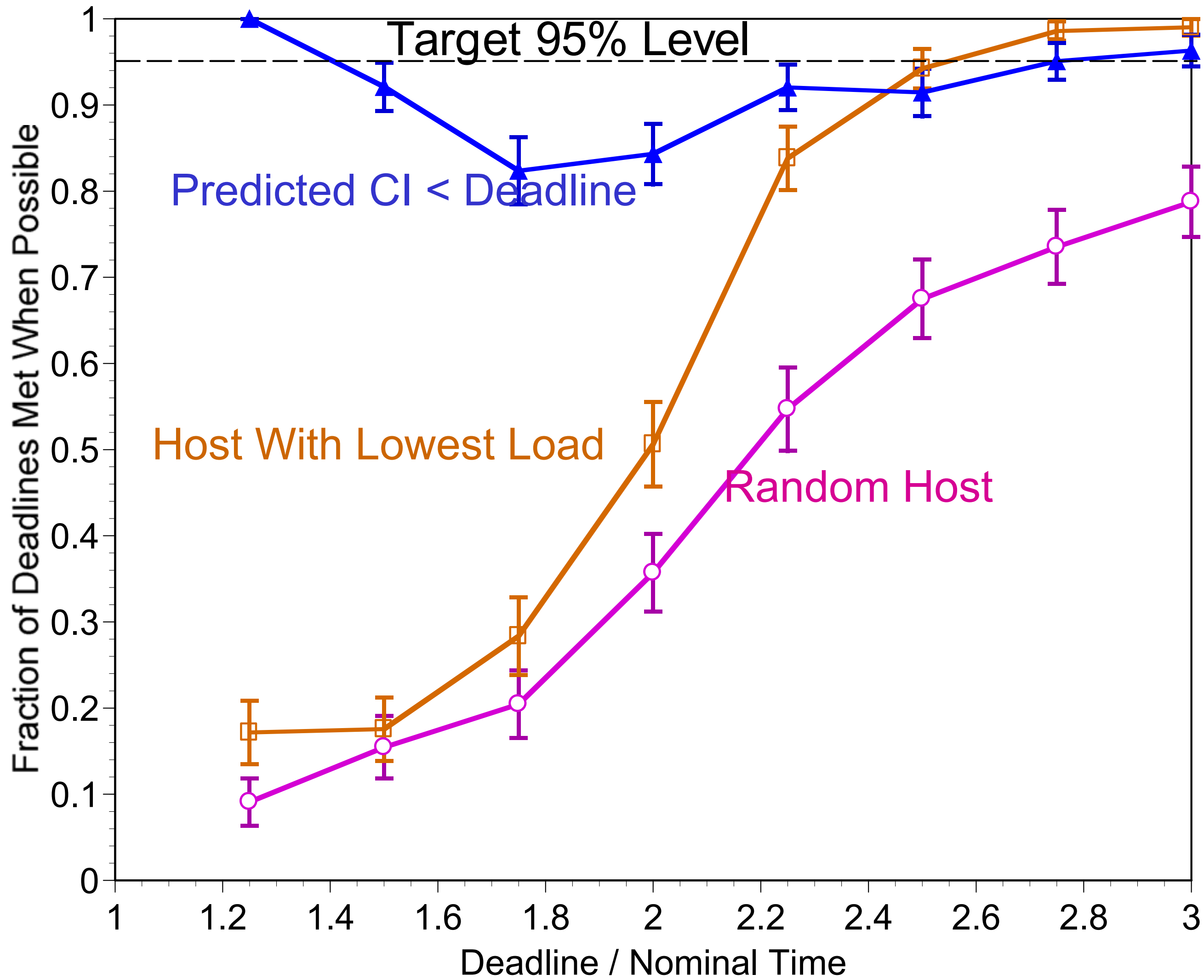


RTSA Results – Probability of Meeting Deadline



16000 tasks

RTSA Results – Probability of Meeting Deadline When Predicted



16000 tasks

RTSA Results

- Application supplies scheduling problem
  - Task size, deadline, and confidence level
  - Task is compute-bound *(current limit)*
- RTSA returns solution
  - Host where task is likely to meet deadline
  - Prediction of running time on that task
- Based on running-time advisor predictions
- Effective
  - Statistically rigorous randomized evaluation

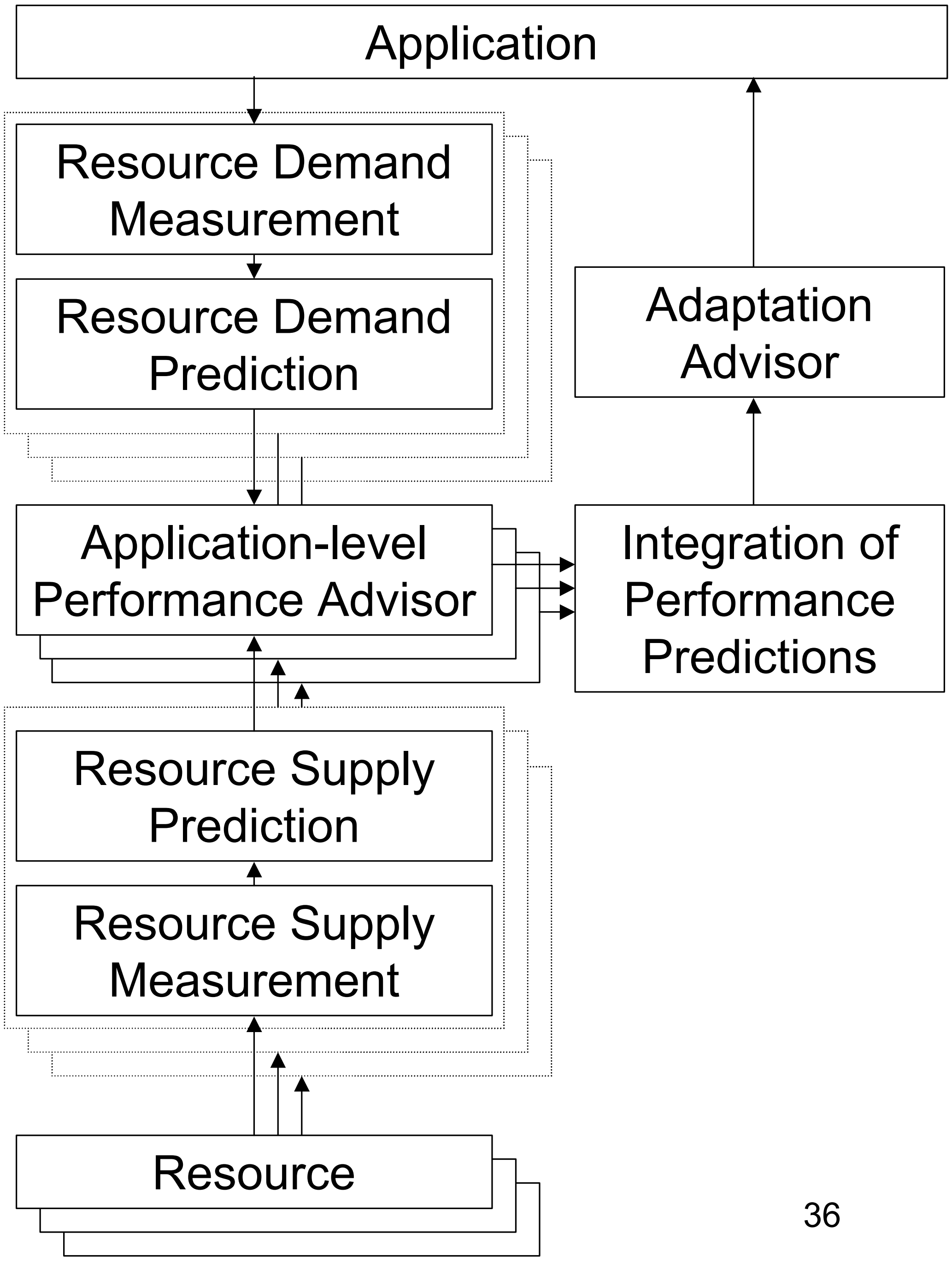
(Study in review)

The Holy Grail

Shared resources scalably provide appropriate measurements and predictions of supply to all comers

Individual applications measure and predict their resource demands

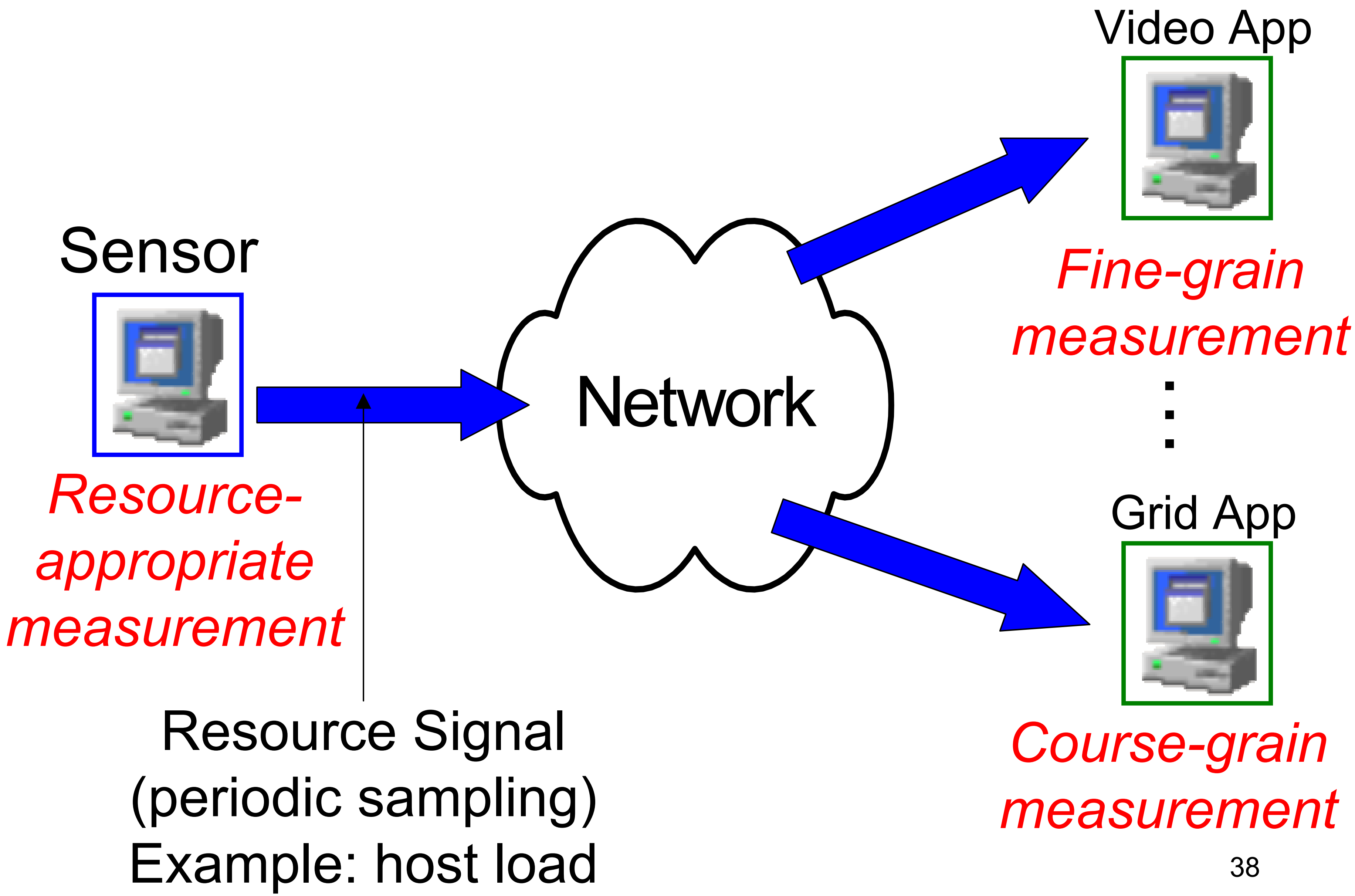
Advisors help applications pursue high-level goals, competing with others



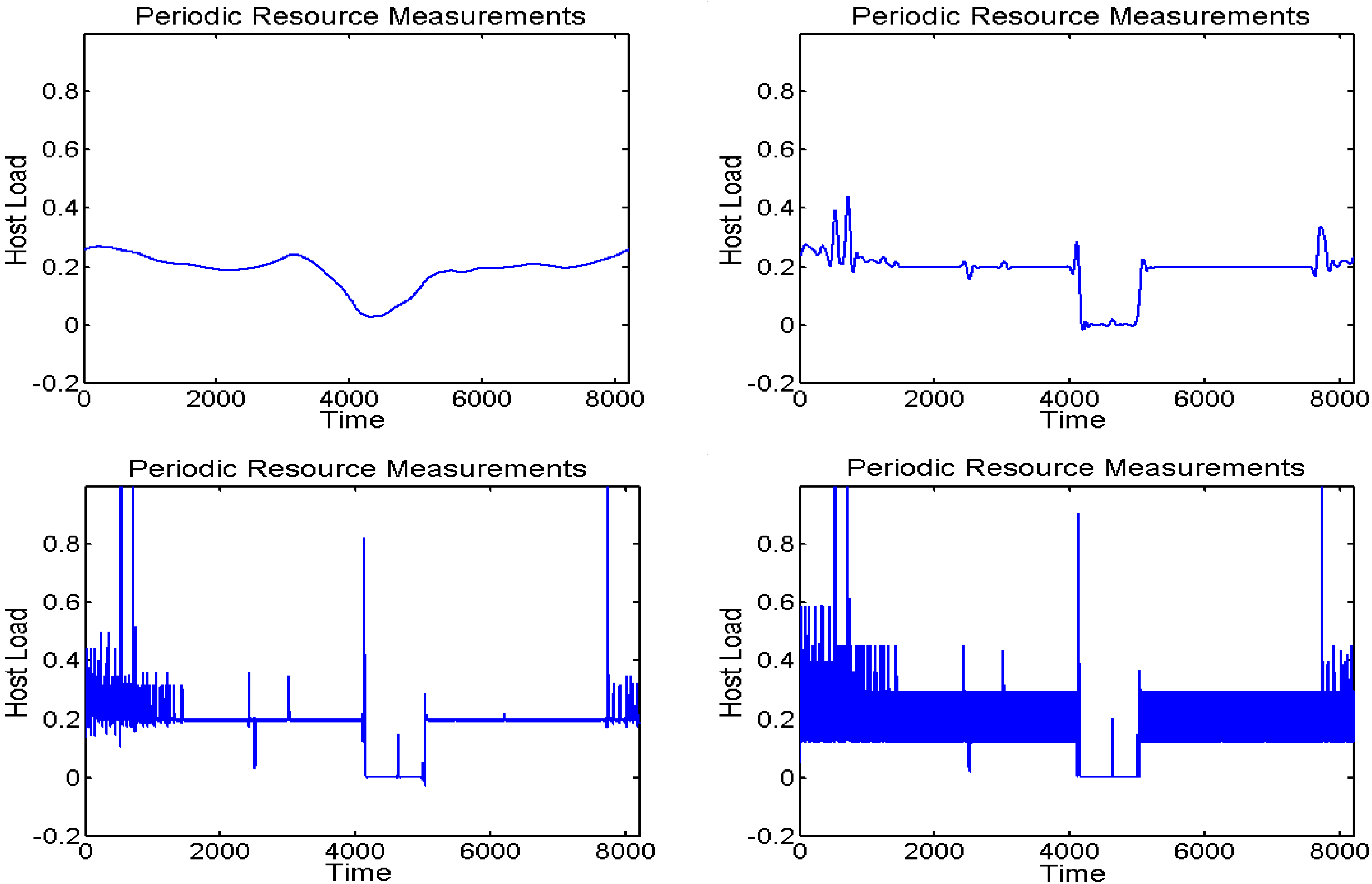
Current work

- Virtualized Audio (with Dong Lu)
- Wavelet-based techniques (with Jason Skicewicz) [HPDC 01]
    - Scalable information dissemination, compression, analysis, prediction
- Network prediction
    - Sampling theory and non-periodic sampling
    - Nonlinear predictive models
    - Minet user-level network stack
- Relational approaches (with Beth Plale and Dong Lu)
    - Grid Forum Grid Information Services RFC [GWD-GIS-012-1]
- Better scheduler models (with Jason Skicewicz)
- Windows monitoring and data reduction (with Praveen Paritosh, Michael Knop, and Jennifer Schopf)
- Application prediction
    - Activation trees
- Clusters for Interactive Applications (with Ben Watson and Brian Dennis)

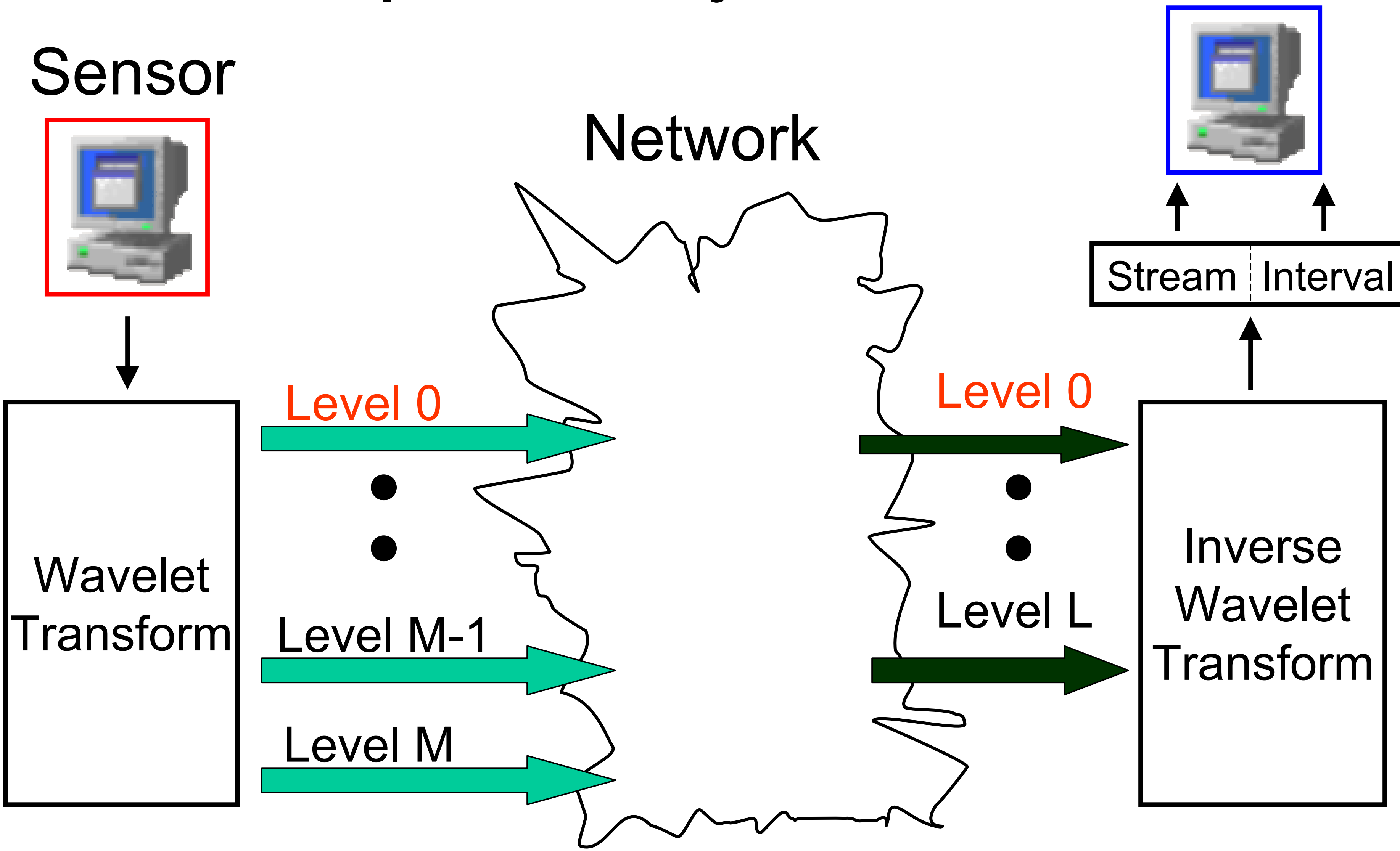
The Tension


Video App
Sensor
Network
Fine-grain measurement
Resource-appropriate measurement
Grid App
Course-grain measurement
Resource Signal
(periodic sampling)
Example: host load

Multi-resolution Views Using 14 Levels

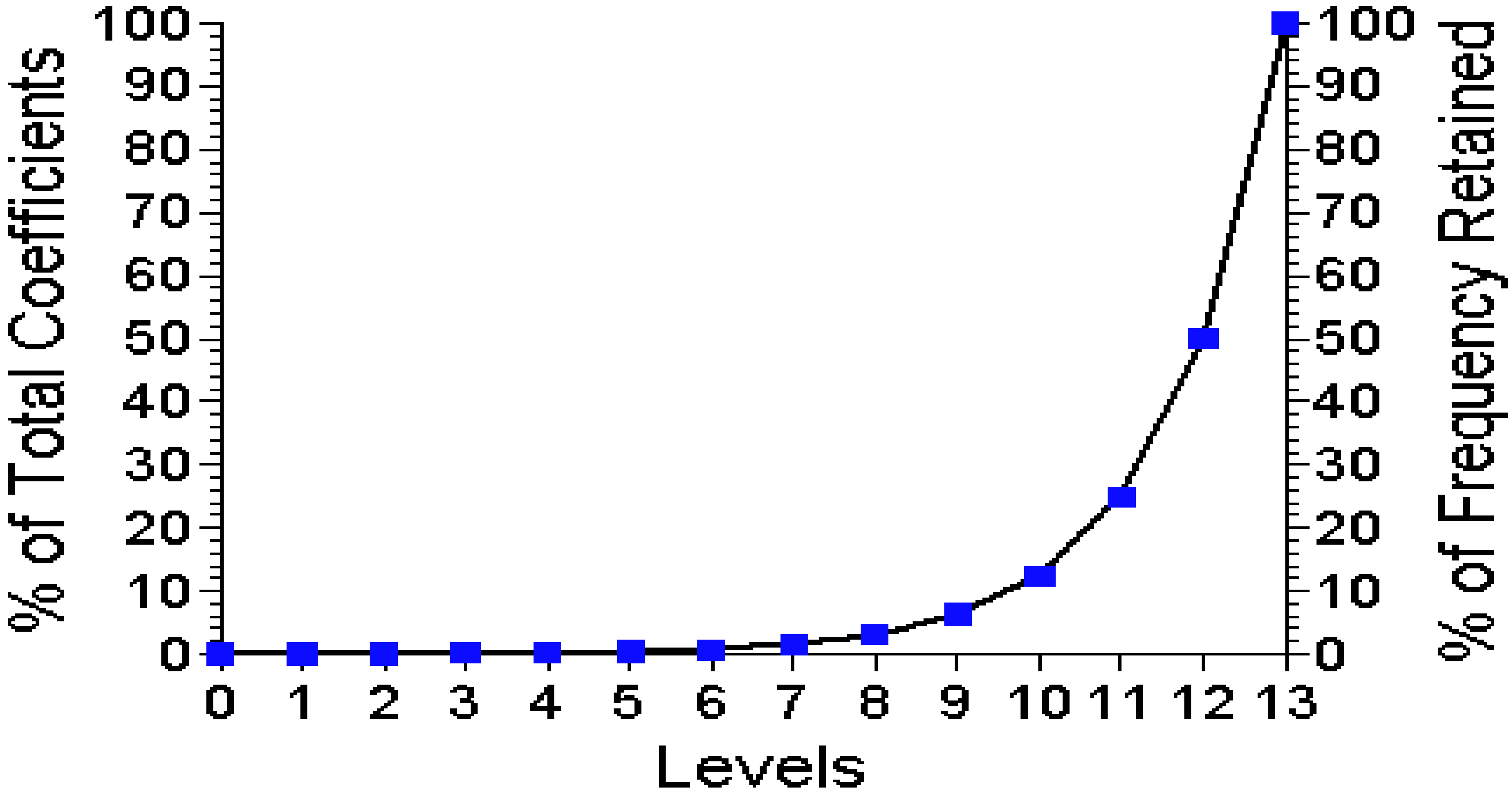


Proposed System



Application receives levels based on its needs

Wavelet Compression Gains, 14 Levels



Typical appropriate number of
levels for host load, error < 20%

For More Information

- http://www.cs.northwestern.edu/~pdinda
- Resource Prediction System (RPS) Toolkit
  - http://www.cs.northwestern.edu/~RPS
- Prescience Lab
  - http://www.cs.northwestern.edu/~plab
- Load Traces and Playload
  - http://www.cs.northwestern.edu/~pdinda/LoadTraces
  - http://www.cs.northwestern.edu/~pdinda/LoadTraces/playload