Dynamic Processor Scheduling with Client Resources for Fast Multi-resolution WWW Image Browsing

Daniel Andresen, Tao Yang, David Watson,
Athanassios Poulakidas

Department of Computer Science

University of California, Santa Barbara

Outline

- Alexandria Digital Library
- Scalability Issues
- Wavelets-based multi-resolution image browsing
- Task partitioning & scheduling
- Analysis
- Experimental Results
- Conclusion

Alexandria Digital Library Project (ADL)

- One of 6 national digital library projects
- Focus on geospatially mapped data
- Potentially enormous holdings - millions of satellite and aerial photos, maps, other holdings.
- Petabyte levels of storage requirements
- Content & metadata access via WWW
- Online system currently in beta-testing

Scalability Issues

Problems:

- **Network bottleneck**

  Not feasible to transmit large images over Internet

- **Server bottleneck**
  - Popular sites receiving millions of hits (Alta Vista at 29M hits/day)
  - ADL queries computationally demanding

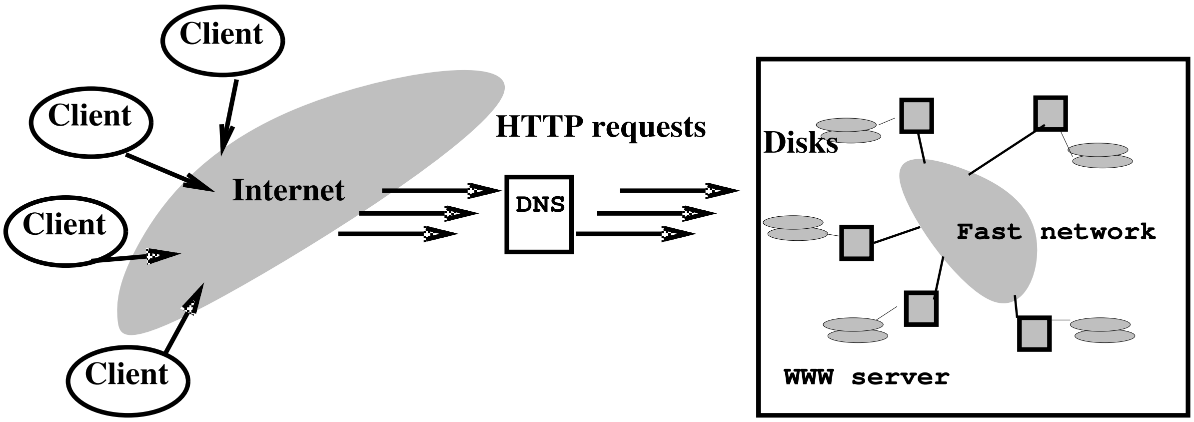
Solutions:

- Compact storage, progressive image delivery
- Multiprocessor support

Paper focus

- Demonstrate multiprocessor server scheduling with client resources to improve performance

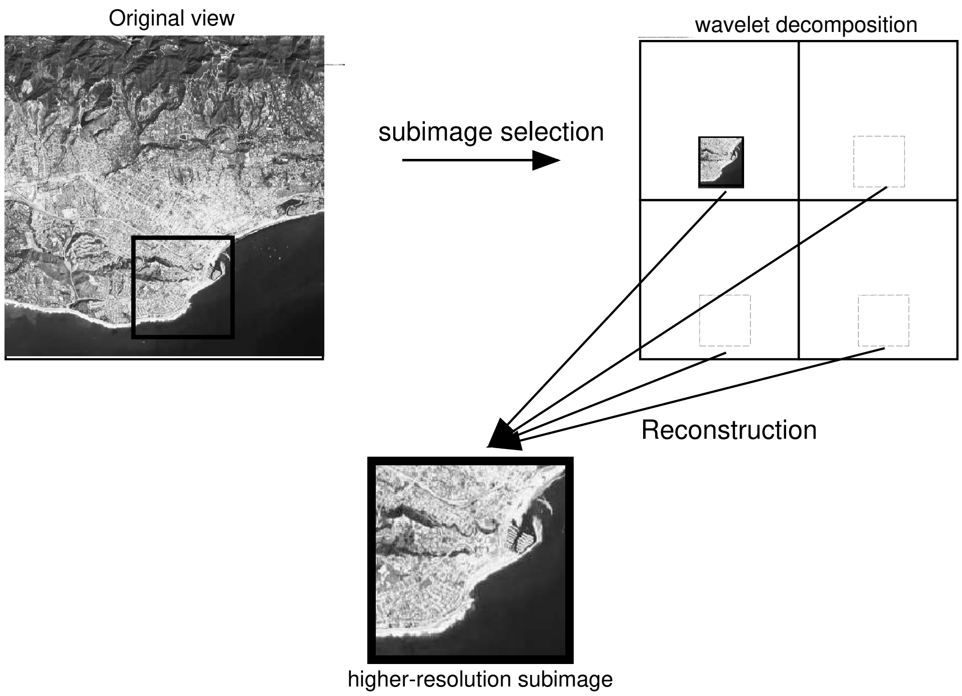
Server Architecture



Assumptions:

- **Server:** fully connected; network file system; distributed scheduler; local disks for parallel I/O
- **Client:** varying capabilities (PDAs to Pentium Pro's)
- **Requests:** file fetches, database queries, image browsing on large files

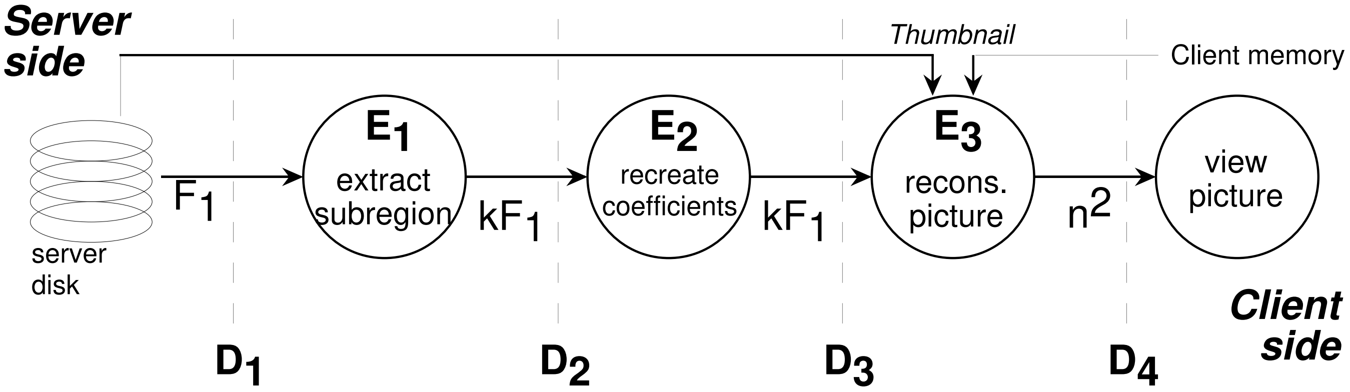
Progressive Image Browsing in the ADL



Reduce the network bandwidth requirement.

1. Deliver an image thumbnail first
2. Deliver additional data for higher resolution.
3. Magnify an image subregion by performing reverse wavelet transform.

Wavelets Computational Model



Four steps

- Extract compressed wavelet coefficients
- Recreating the coefficients
- Reconstructing the pixels
- Viewing the image

Four breakpoints

- D1 - everything on client
- D2 - decompression/reconstruction on client
- D3 - reconstruction on client
- D4 - everything on server

Distributed Scheduling Algorithm

For each request:

Estimate the client CPU load and speed.

For each available server node s

    Estimate loads, local/remote disk bandwidth

    For each possible client-server split point k

        Evaluate t_s

    Select split point minimizing t_s.

EndFor

Select the best server node.

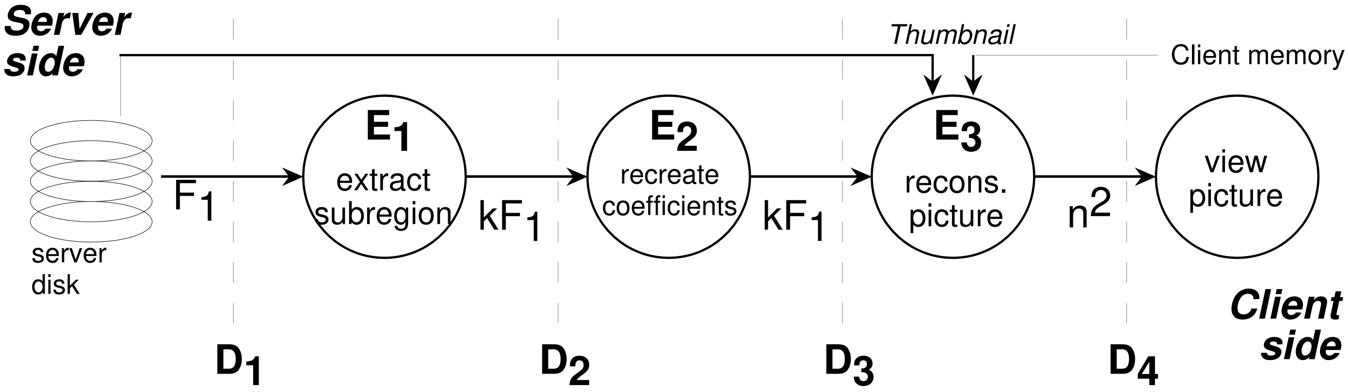
Predict request response time:

$$t_s = t_{redirection} + t_{data} + t_{server} + t_{net} + t_{client}$$

based on loads, bandwidths, file locality, and request analysis.

Analysis on Effectiveness

Analyze expected *maximum sustained requests per second (MRPS)*, *redirection rates*, and *split points.*



Derive expected response time H_1 for **D1:**

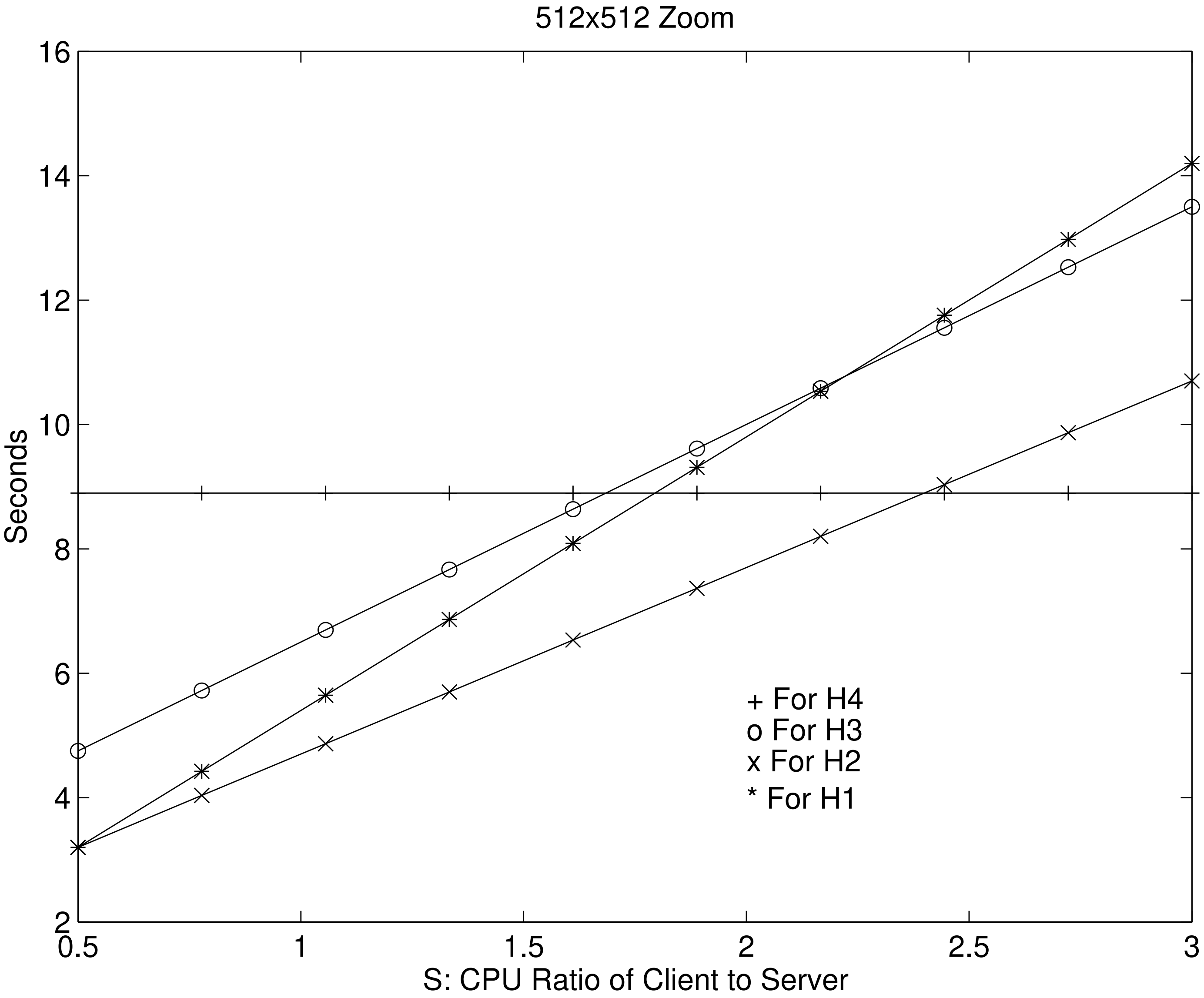
$$H_1 = f + rgF_1 + S(E_1 + E_2 + E_3) + F_1/B$$

- f = overhead & file access time
- r = concurrent requests
- $S = \frac{CPU_{\text{server speed}}}{CPU_{\text{client speed}}}$
- B – server-client network bandwidth
- g – file transfer cost

$\ldots H_2$, H_3, and H_4 for D2, D3, and D4.

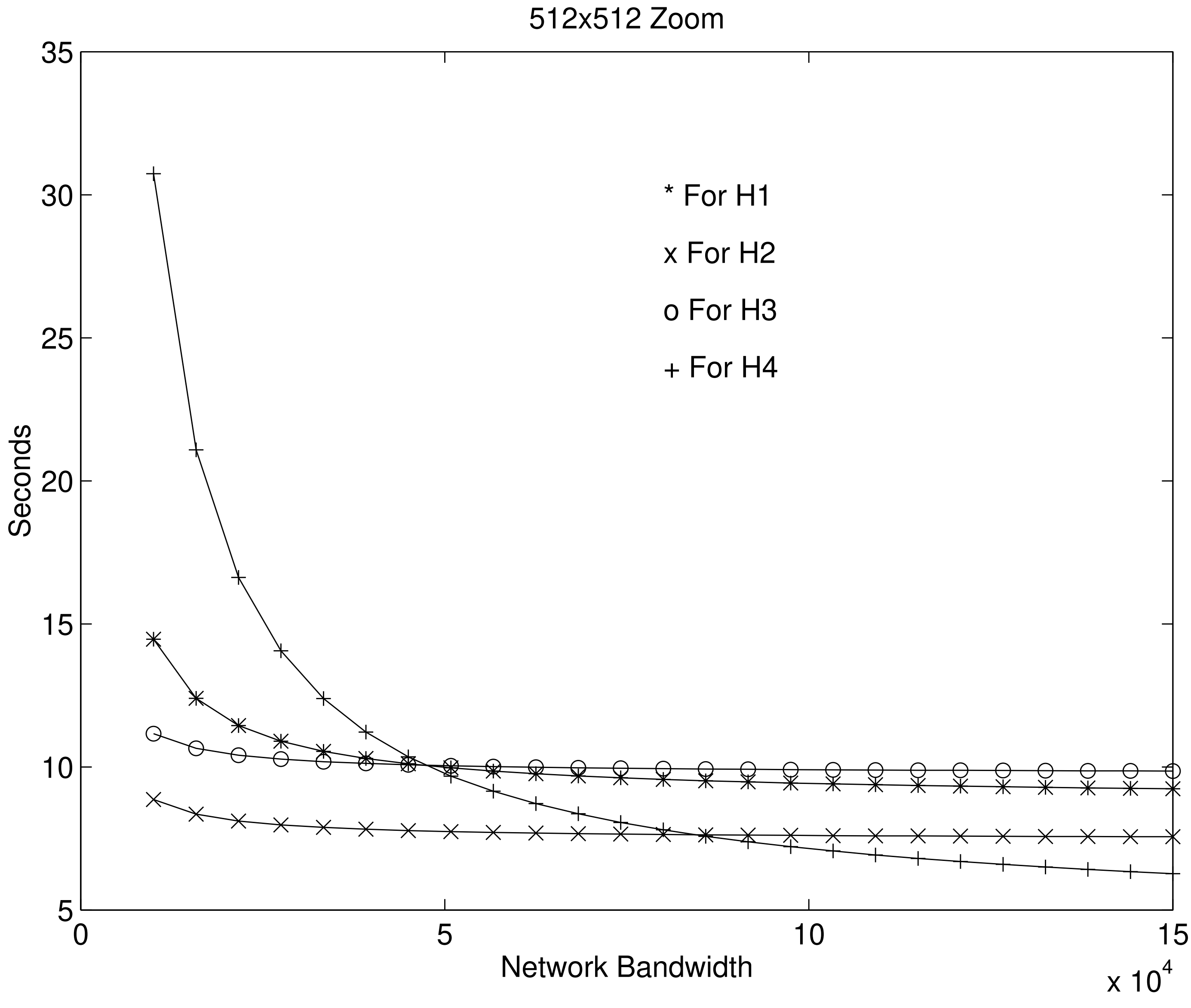
Wavelet split points

Varying client speeds



$$H = min(H_1, H_2, H_3, H_4)$$

Varying network speeds



$$H = min(H_1, H_2, H_3, H_4)$$

Experimental Results

Setting

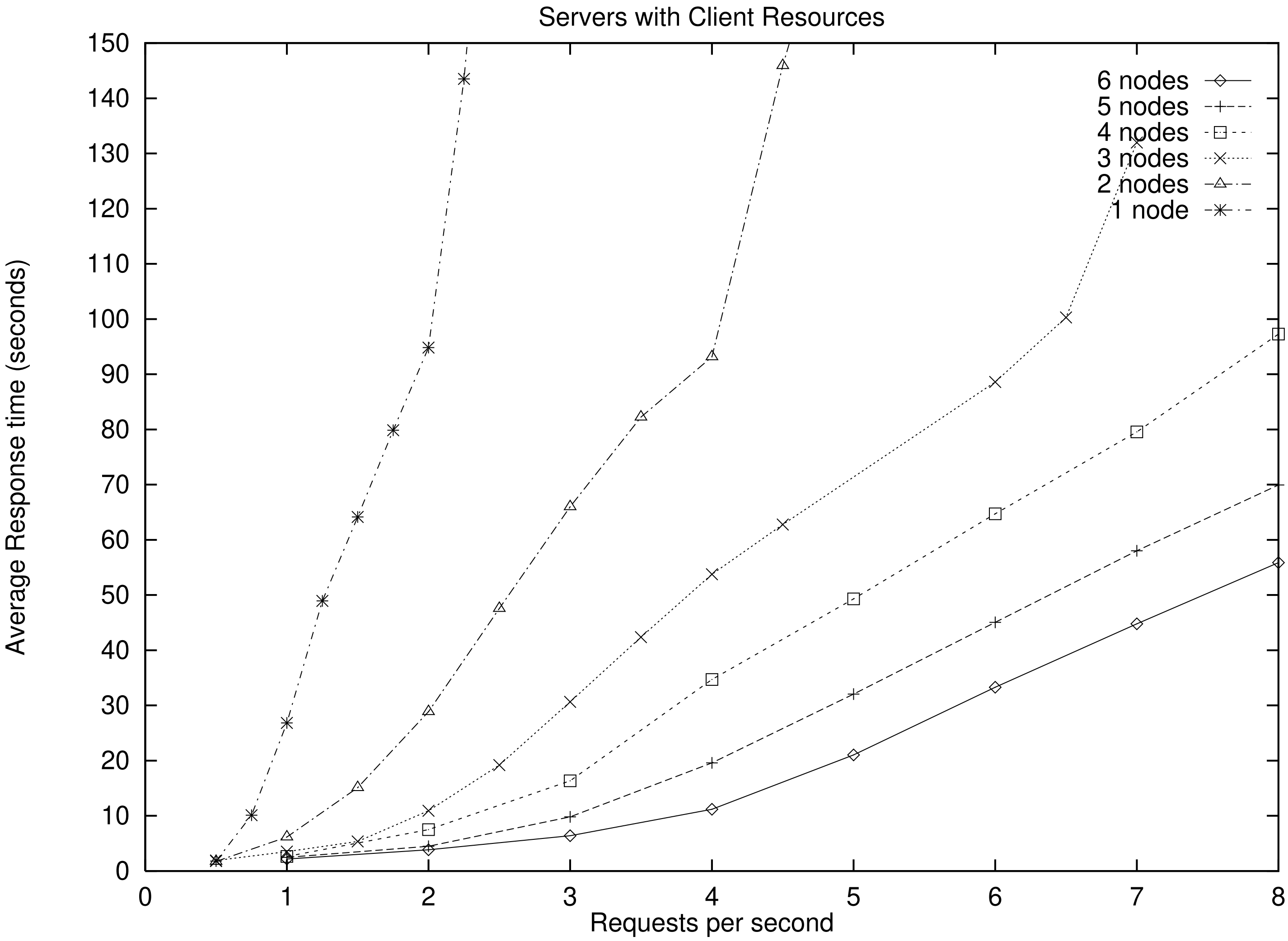
- Server: 6 Sparc 10's on Meiko CS-2, local 1GB disk, fast interconnection network
- Clients: 16-32 Sparc 10's on Meiko & local Ethernet network

Results

- *General results:* Effect of adding servers; Effect of client resources; Load balancing with Hotspots
- *Analytical verification:* Scheduling decisions; Predicted breakpoints

Effect of Multiple Servers

Response time (seconds)



Requests per second

Effect of using Client Resources

Response times (in seconds, 6 nodes.)

RPS	1.0	1.5	2.0	2.5
Without	15.97	57.24	123.76	213.4
With	4.78	5.61	6.45	7.73
Improvement	234%	918%	1818%	2660%

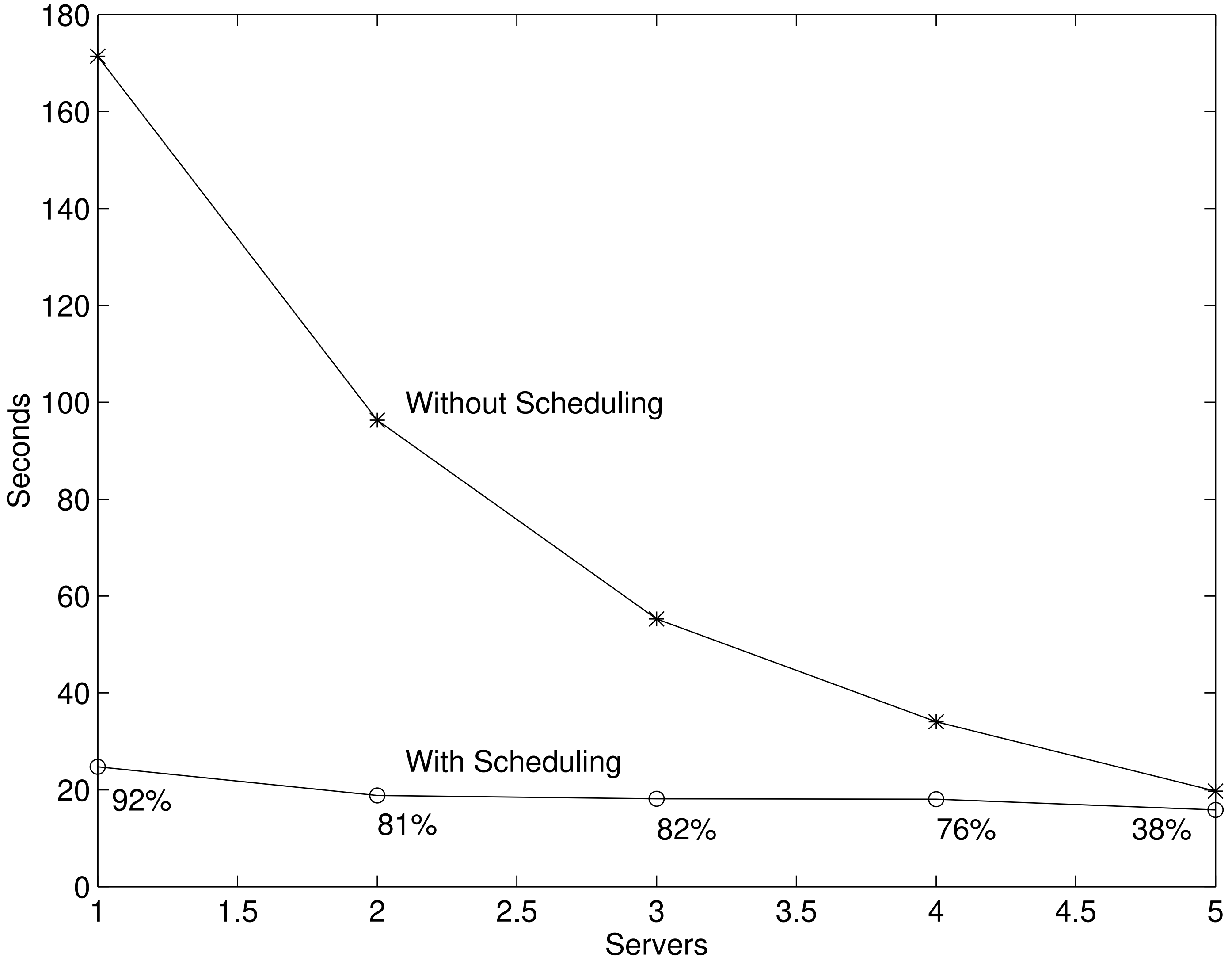
Bursty maximum RPS for wavelets.

	p=1	2	3	4	5	6
Without	0.3	0.6	0.8	1.2	1.4	1.5
With	1.5	3.0	4.5	6.0	7.5	9.0

Average 433% improvement in MRPS

Load balancing with hotspots

Requests are routed to subset of server nodes



Wavelets subimage extraction, 4RPS, % is redirection rate.

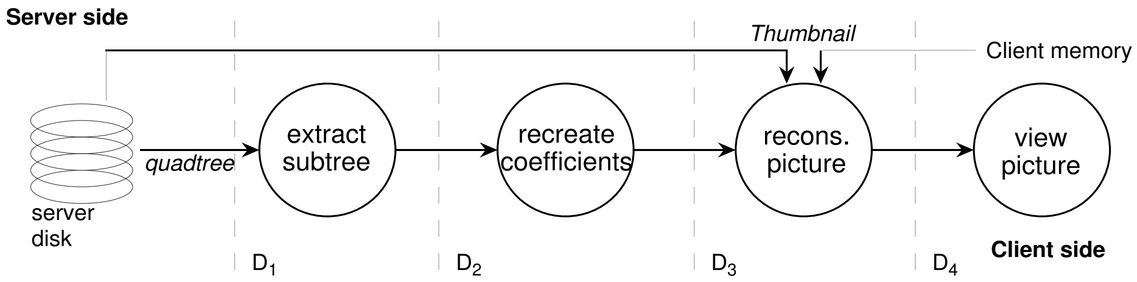
Actual vs. predicted split points

Burst of 18 wavelet requests

Server/Client CPU Speed Ratio

Client-server bandwidth (bytes/sec)	0.25	0.5	1	2	4	10	20	40
100	D2	D2	D2	D2	D2	D2	D2	D2
500	D2	D2	D2	D2	D2	D2	D2	D2
1,000	D2	D2	D2	D2	D2	D2	D2	D2
10,000	D2	D2	D2	D2	D2	D2	**D4**	**D4**
100,000	D2	D2	D2	76	**6**	**D4**	**D4**	**D4**
150,000	D2	D2	D2	77	**D4**	**D4**	**D4**	**D4**
250,000	D2	D2	D2	93	**D4**	**D4**	**D4**	**D4**
500,000	D2	D2	D2	95	**D4**	**D4**	**D4**	**D4**
750,000	D2	D2	D2	88	**D4**	**D4**	**D4**	**D4**
1,000,000	D2	D2	D2	98	**D4**	**D4**	**D4**	**D4**

Predicted D2 region ***Predicted D4 region***



Conclusions & Current Work

Conclusions

- Scheduling using client resources dramatically improves response times.
- Dynamic scheduling is necessary to adapt to varying client capabilities.

Current work

- Generalize scheduling algorithm
- Generalize analytical framework
- Implement system to support dynamically-partitioned WWW applications.