Part II: Comparing Server Performance

Tim Brecht

www.cs.uwaterloo.ca/~brecht

Outline

- Part I: Background
  - Web Server Example: HTTP/1.1
  - Server Architectures
  - Performance Evaluation
- **Part II: A Flavour of some Current Research**
  - **Performance of Different Server Architectures**
  - Improving Operating System Support for I/O Centric Servers (if time permits)
  - **Possible Avenues for Future Research**

Outline

- **Part II: A Flavour of some Current Research**
  - Performance of Different Server Architectures
  - Improving Operating System Support for I/O Centric Servers (if time permits)
  - Possible Avenues for Future Research

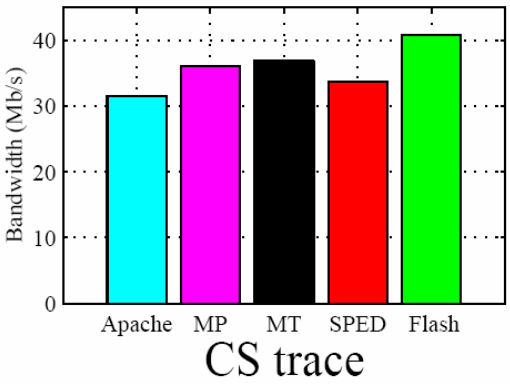
Recall: Some Server Architectures

- Multi-Thread/Process: one thread/process per conn
  - (MT/MP) **Apache, Knot** [apache.org, von Behren et 03]
- Single Process Event Driven [www.zeus.com]
  - (SPED) **Zeus, Original Harvest/Squid** [Wessels, 96]
  - Asymmetric Multi-Process Event Driven
  - (AMPED) **Flash** [Pai et al, 99]
- One copy per CPU [Zeldovich et al, 03]
  - (N-Copy) **? Rock Web Server ?** [accoria.com]
- SYmmetric Multi-Process Event Driven
  - (SYMPED & Shared-SYMPED) **userver** [UW:Brecht et,]
- Hybrid: Staged Event Driven Architecture / Pipelined
  - (SEDA) **Haboob, WatPipe** [Welsh et 01, Pariag et 07]

Some Research History: Threads vs Events

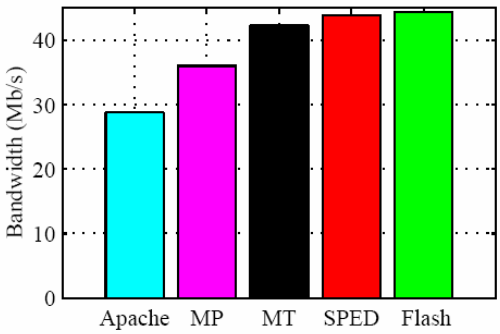
- Duality of OS Structures [Lauer & Needham, 78]
- Why Threads are a Bad Idea [Ousterhout, 96]
- SEDA => Events are Best [Welsh et al, 01]
- Event Driven Programming for Robust Software [Dabek et al, 02]
- Why Events are a Bad Idea [von Behren et al, 03]
- Capriccio: Scalable Threads ... [von Behren et al, 03]
- Multiprocessor Support for Events [Zeldovich et al, 03]

Previous: Flash [from Pai et al, 1999]



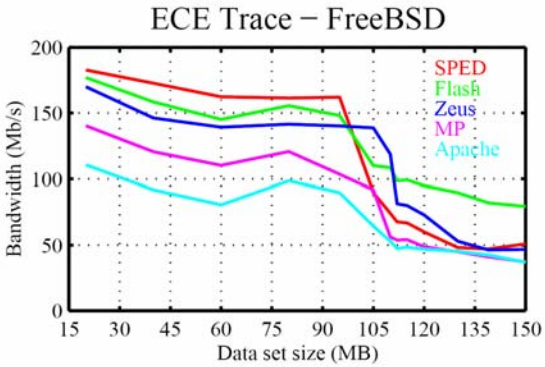
CS trace

Previous: Flash [from Pai et al, 1999]



Owlnet trace

Previous: Flash [from Pai et al, 1999]

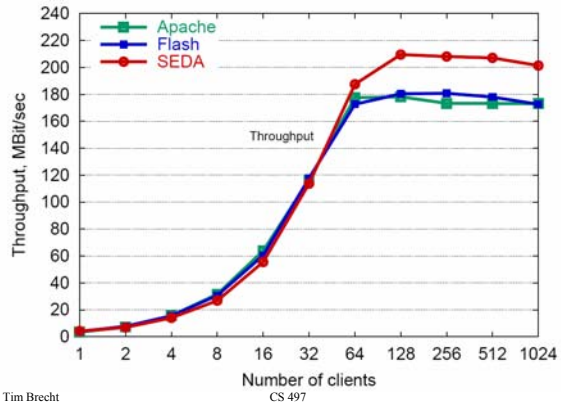


Conclusions

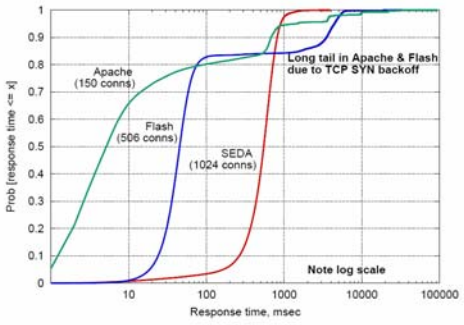
- Amped is best (Flash) [Pai et al. 1999]

Amped ~> SPED > MT/MP > Apache
Flash Flash Flash

Previous: Haboob [from Welsh et al, 2001]



Previous: Haboob [from Welsh et al, 2001]



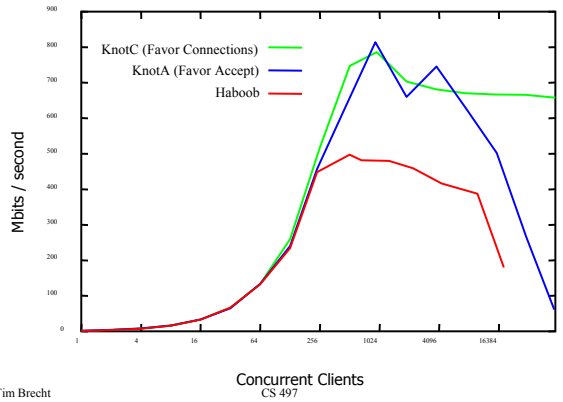
	SEDA	Flash	Apache
Mean RT	547 ms	665 ms	475 ms
Max RT	3.8 sec	37 sec	1.7 minutes

Conclusions

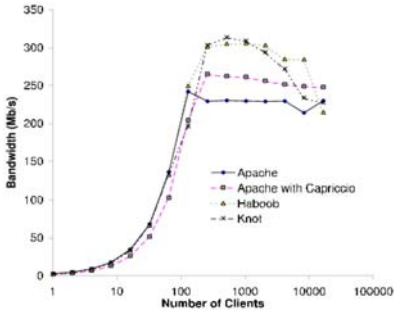
- Amped is best (Flash) [Pai et al. 1999]
- Events are best (SEDA/Haboob) [Welsh et al. 2001]

SEDA > Amped ~> SPED > MT/MP > Apache
Haboob Flash Flash Flash

Previous: Knot [from von Behren et al 03a]



Previous: Knot [from von Behren et al 03b]



Conclusions

- Amped is best [Pai et al. 1999]
- Events (using SEDA) are best [Welsh et al. 2001]
- Threads are best (using Capriccio) [von Behren et al 03]

MT >= SEDA > Amped ~> SPED > MT/MP > Apache
Knot Haboob Flash Flash Flash
Capricco

Conclusions

- Amped is best [Pai et al. 1999]
- Events (using SEDA) are best [Welsh et al. 2001]
- Threads are best (using Capriccio) [von Behren et al 03]

MT >= SEDA > Amped ~> SPED > MT/MP > Apache
Knot Haboob Flash Flash Flash
Capricco

Outline

- Part II: A Flavour of some Current Research
  - **Performance of Different Server Architectures**
  - Improving Operating System Support for I/O Centric Servers (if time permits)
  - Possible Avenues for Future Research

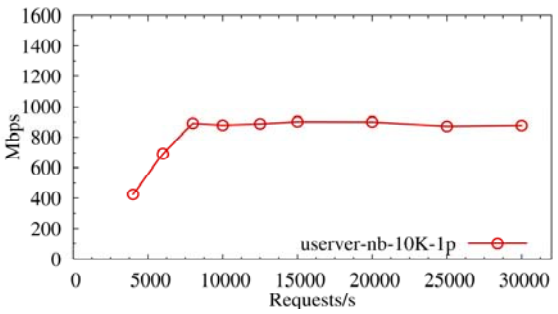
Comparing Performance, EuroSys 2007

- Compare different modern server architectures:
  - Thread-based or Thread-per-Connection or (MT)
    Capriccio / Knot
  - Event-Driven (SYMPED/Shared-SYMPED)
    userver
  - Pipeline-based / Hybrid (Threads and Events)
    WatPipe
    - similar to SEDA/Haboob but in C++; not Java
    - no controllers, no tuning of controllers

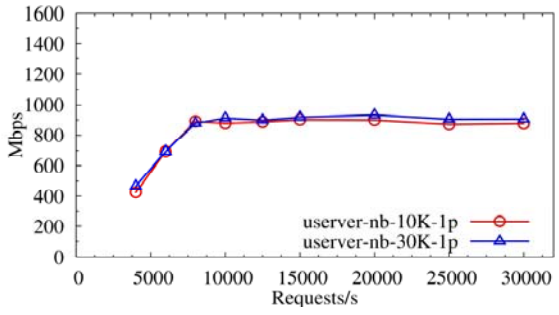
Comparing Performance, EuroSys 2007

- Fair comparison, well configured, well tuned servers
  (avoid using Java, as it could be slow)
- Verification: ensure service for all request sizes
  - servers often timing out on large requests
- 3.2 GB file set as in Capriccio and SEDA papers
- **`httperf`** for overload (using client timeouts = 15 s)
- Workload simulates think times
  - forces servers to scale to large numbers of clients (10's of thousands)

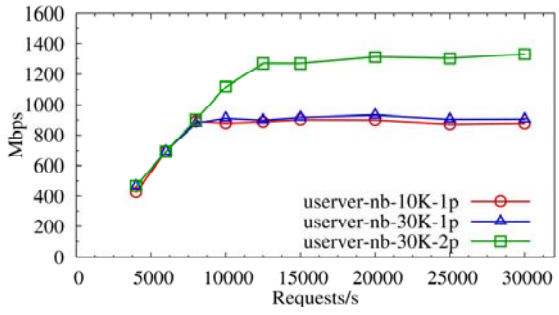
SYMPED (userver)



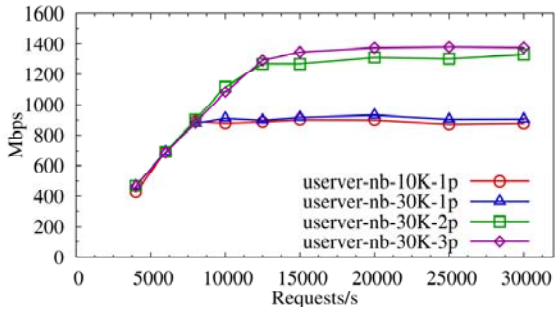
SYMPED (userver)



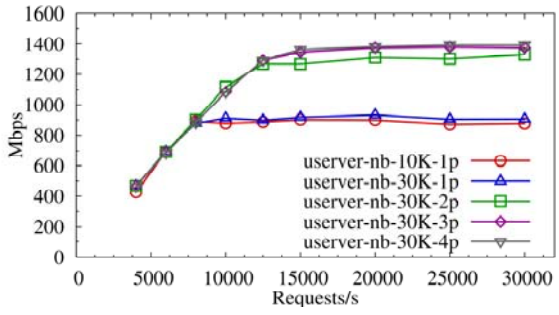
SYMPED (userver)



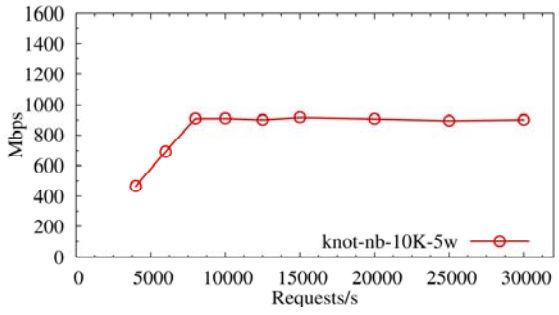
SYMPED (userver)



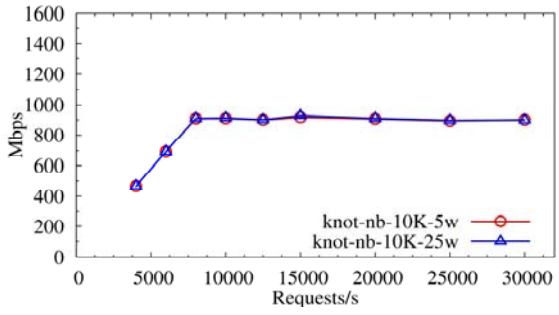
SYMPED (userver)



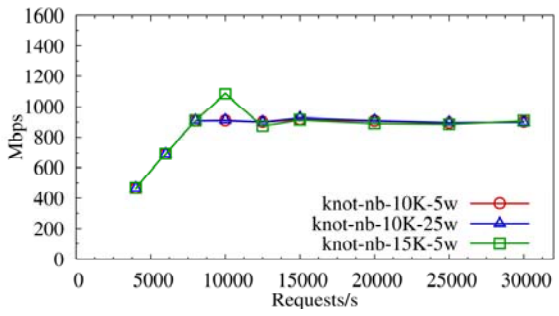
Thread-per-connection (Knot)



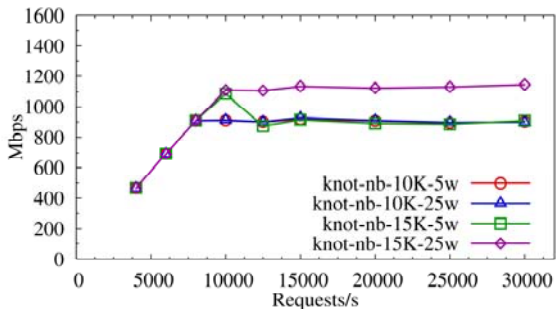
Thread-per-connection (Knot)



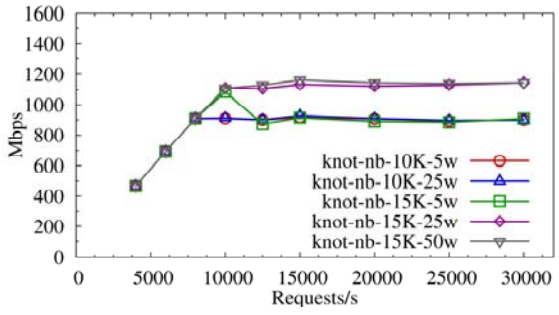
Thread-per-connection (Knot)



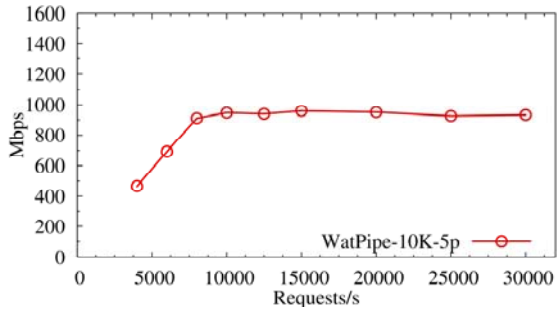
Thread-per-connection (Knot)



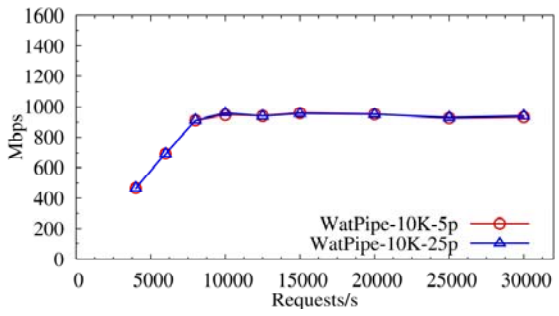
Thread-per-connection (Knot)



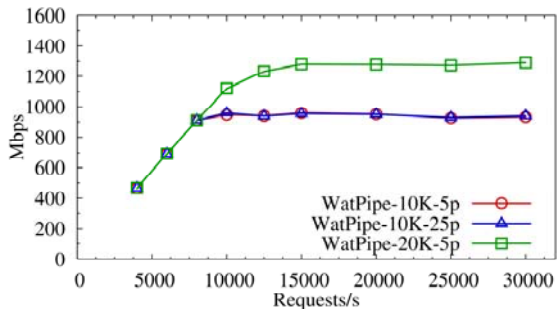
Hybrid / WatPipe



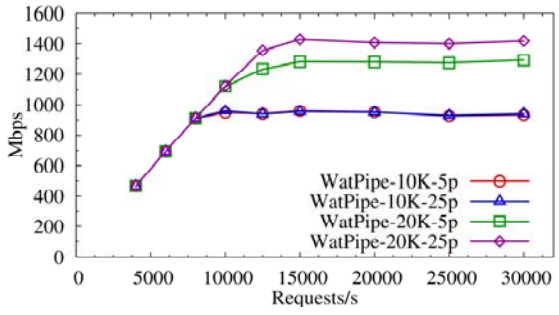
Hybrid / WatPipe



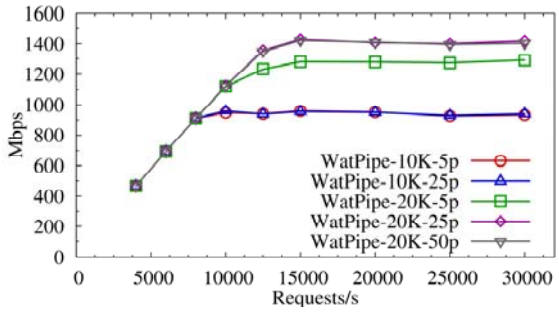
Hybrid / WatPipe



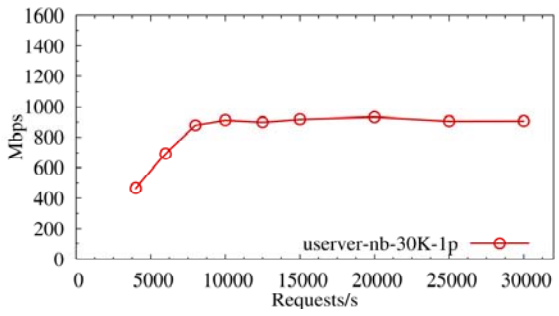
Hybrid / WatPipe



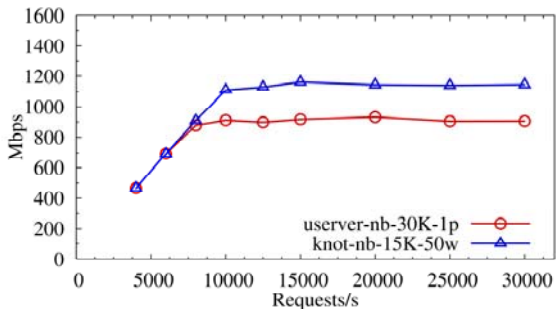
Hybrid / WatPipe



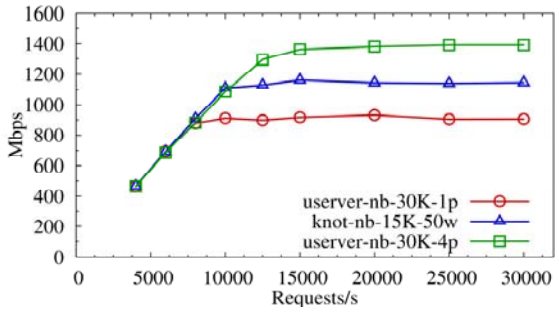
Comparing the Best of Each Server



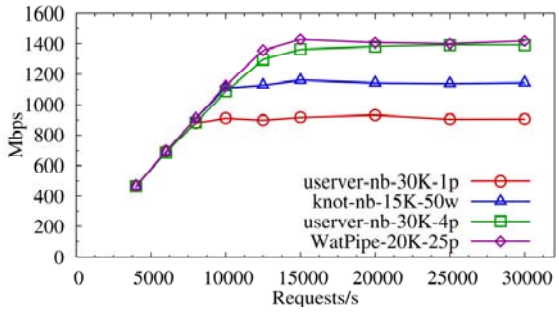
Comparing the Best of Each Server



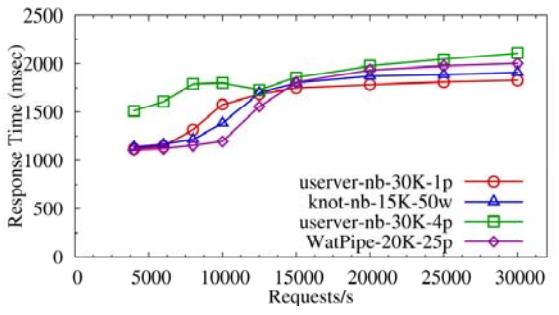
Comparing the Best of Each Server



Comparing the Best of Each Server



Comparing the Best of Each Server (RT)



SUMMARY

- SYMPED (userver) ~= Pipelined (WatPipe) >
- MT (Knot/Capriccio) >=
- SEDA (Haboob) >
- AMPED (Flash) ~>
- SPED (Flash) >
- MT/MP (Flash) >
- Apache

SUMMARY

- SYMPED (userver) ~= Pipelined (WatPipe) >
- MT (Knot/Capriccio) >=
- SEDA (Haboob) >
- AMPED (Flash) ~>
- SPED (Flash) >
- MT/MP (Flash) >
- Apache

Really? – Under which workloads?
How well were the servers tuned?
What about multiprocessor environments?

Epilogue

- BUT ...
  - Threads Cannot Be Implemented As a Library, Hans Boehm, Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (PLDI), June 2005, pp. 261-268.

SO CLEARLY EVENTS ARE BEST ☺

Outline

- Part II: A Flavour of some Current Research
  - Performance of Different Server Architectures
  - Improving Operating System Support for I/O Centric Servers (if time permits)
  - **Possible Avenues for Future Research**

Some Ongoing and Future Research

- **Problem**: Tuning is a huge pain
- **Research**: Automatic and dynamic tuning
  - Existing servers use static configuration parameters
  - Dynamically adjust parameters (e.g.,):
    - number of connections
    - number of processes/workers/helpers
  - Goal: start a web server, it configures itself
    - Processors, Memory, Disks, Workload
    - Performance sufficiently close to manual tuning
  - Probably requires better understanding & models

Some Ongoing and Future Research

- **Problem**: 80-90% of CPU time is spent in kernel
- **Research**: Improve Application / OS Interaction
  - Improve operating system implementation
  - Change application to better use existing interfaces
  - Add new system calls and modify apps to use them
  - Unable to utilize 10 Gbps Ethernet cards effectively
    - Can utilize 8 x 1 Gbps ethernet cards better than 1 x 10 Gbps
    - 100 Gbps cards have been demonstrated

Some Ongoing and Future Research

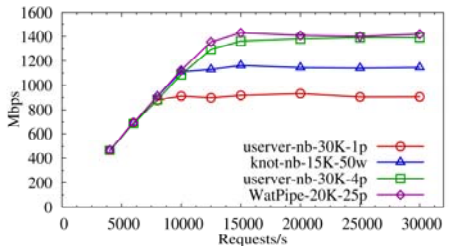
- **Problem**: Most research: Single CPU, static requests
  Most servers: Multi-processor, dynmamic
- **Research a)**: Effective utilization of multi-processors
- **Research b)**: Improving dynamic requests
- **Research c)**: Combining the two
  - improve communication: web & app servers
  - # of app servers, how much cpu power, autotuning

  E.g.: 4 cpus, how many web server processes,
  how many app server processes, does it matter
  where they run?

Some Ongoing and Future Research

- **Problem**: When to stop improving performance?
- **Research :** How fast could a server possibly run?

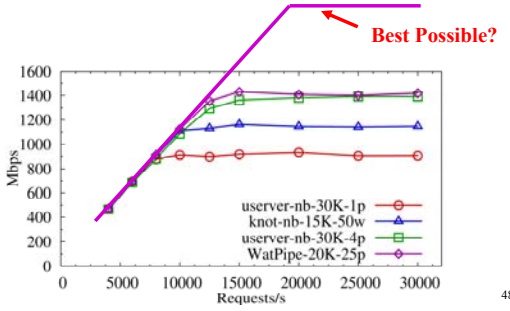
Best Possible?



Some Ongoing and Future Research

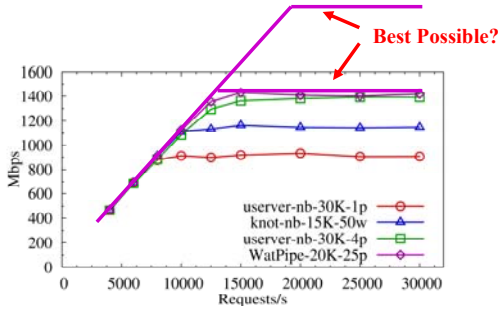
- **Problem**: When to stop improving performance?
- **Research :** How fast could a server possibly run?

Best Possible?



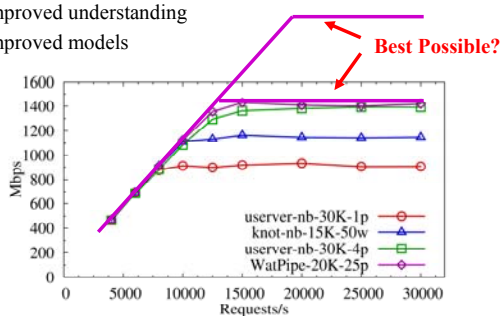
Some Ongoing and Future Research

- **Problem**: When to stop improving performance?
- **Research :** How fast could a server possibly run?



Some Ongoing and Future Research

- **Problem**: When to stop improving performance?
- **Research :** How fast could a server possibly run?
  - improved understanding
  - improved models



THE END