OpenACS and EuroTcl 2023

NaviServer 5.0

Univ.-Prof. Dr. Gustaf Neumann

Vienna University of Economics and Business
Information Systems and New Media

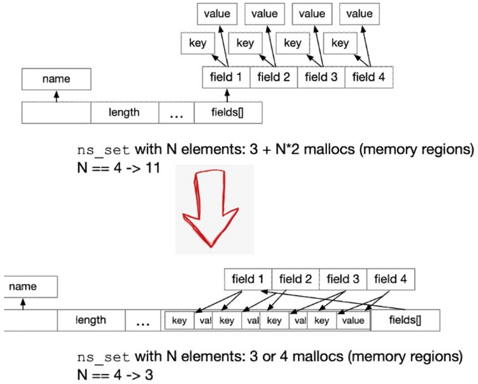
WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EQUIS ACCREDITED AACSB ACCREDITED AMBA ACCREDITED

JULY 20, 2023

Overview

- What's new?
  - New Licence
  - Tcl9 compatibility
  - Significant code changes
  - New Major version number
- Reforms
  - Improved memory locality for ns_set
  - Persistent connections for ns_http
  - Removed usage of double-checking lock pattern
  - Clustering
- Want's next?
  - HTTP/2, HTTP/3?
  - More protocols



Why a new major number?

- NaviServer Releases:
  - 4.99.0 … 4.99.26
  - ”Running out of fingers and toes”
    (Citation of Linus Torvalds, when Linux stepped up to 3.20)
- New License:
  - Upgrade from Mozilla Public License Version 1.1 + GPL
  - to Mozilla Public License (MPL) 2.0
- Tcl 9:
  - Lifting various restrictions (32-bit signed integers -> 64-bit)
  - Substantial code changes in NaviServer necessary to make use of new capabilities
  - Release of NaviServer 5 will be after the release of Tcl 9
- New Features
  - A few set of changes cherry-picked on the next slides
  - Improved crypto functionality: E.g. support for Argon2 (winner of the 2015 Password Hashing Competition, defined by RFC 9106)
  - NaviServer 5.0 works with Tcl 8.6 and Tcl 9 (regression test with GitHub workflows)
- EOL NaviServer 4.99.*?
  - No, bugfixes still in the 4.99 branches, leading to 4.99.27 etc
  - Many NaviServer / OpenACS user are conservative

ns_set reform (1/3)

- What is an ns_set:
  - NaviServer data structure for the Tcl programmer
  - Like a Tcl dict, supporting duplicate keys, having names
  - Predates Tcl dict significantly (before 2000)

- Used for:
  - HTTP header fields
  - Configuration values
  - SQL tuples
  - …

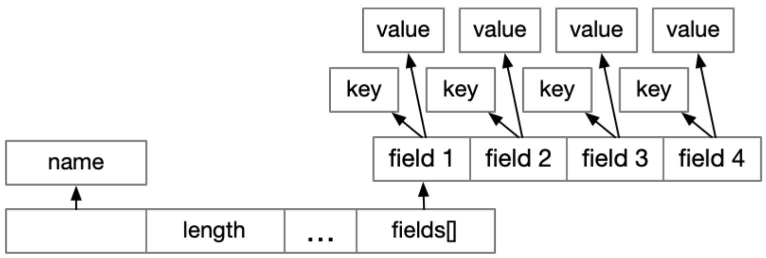
- Example:
  - SQL query, returning 20 attributes, 1000 Tuples, e.g.: "`select * from acs_objects limit 1000`"
  - 43.000 malloc/free operations (*1000*(3 + 20*2)*)
  - This is for OpenACS installations a small query, many return 100K tuples or more

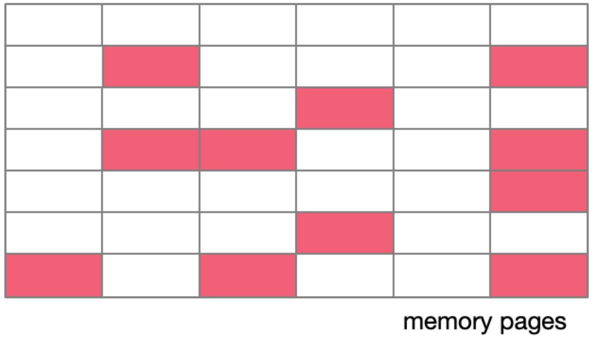
ns_set reform (2/3)

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Memory pages

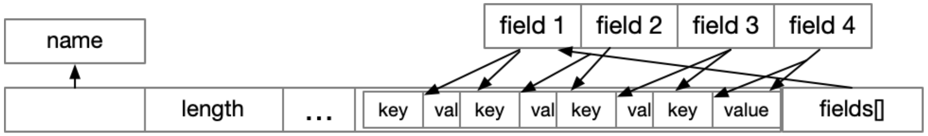
Old:





`ns_set` with N elements: 3 + N*2 mallocs (memory regions)
N == 4 -> 11

New:



`ns_set` with N elements: 3 or 4 mallocs (memory regions)
N == 4 -> 3

Improved memory locality

- Based on Tcl_DStrings
- More CPU-cache hits, improved performance
- Less memory consumption
- Less mutex locks

CPU Cache management

- Changes in pages require refetch
- Multi-threading: refetch per thread
- Especially expensive with NUMA architectures
- Memory access might differ by a factor of 5 or more

ns_set reform (3/3)

Quick test:

- Running sample query (1000 tuples a 20 attributes) in
- 1..30 threads
- Xeon Gold 6226R CPU @ 2.90GHz, 32 cores, hyper-threading enabled

Before (classical ns_set with many mallocs):

```
threads 1 total 4606.787 ms avg 3285.25 ms
threads 5 total 4595.358 ms avg 3493.07 ms
threads 10 total 4804.193 ms avg 3755.93 ms
threads 20 total 6279.524 ms avg 4569.16 ms
threads 30 total 8966.427 ms avg 6618.58 ms
```

After reform (using one Tcl_DString per tuple):

```
threads 1 total 4524.645 ms avg 3242.54 ms
threads 5 total 4251.266 ms avg 3450.09 ms
threads 10 total 4656.795 ms avg 3665.31 ms
threads 20 total 5934.105 ms avg 4671.38 ms
threads 30 total 7384.591 ms avg 5642.76 ms
```

E.g. with 30 threads, the total time improved by 17%.... with a smaller RSS.

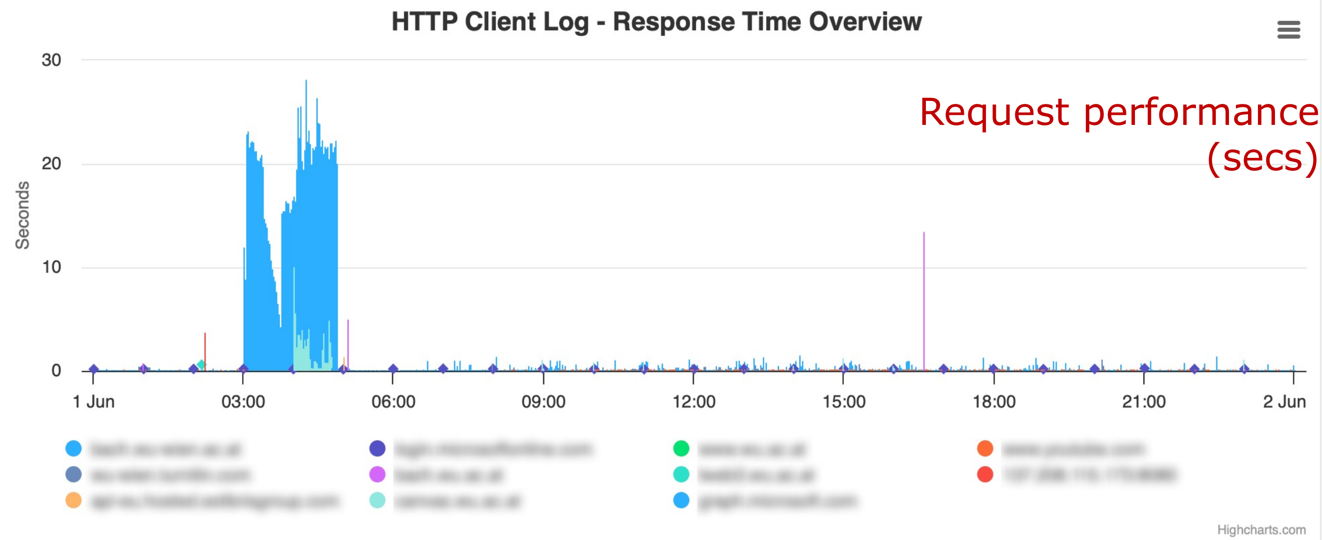
ns_http reform (1/3)

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

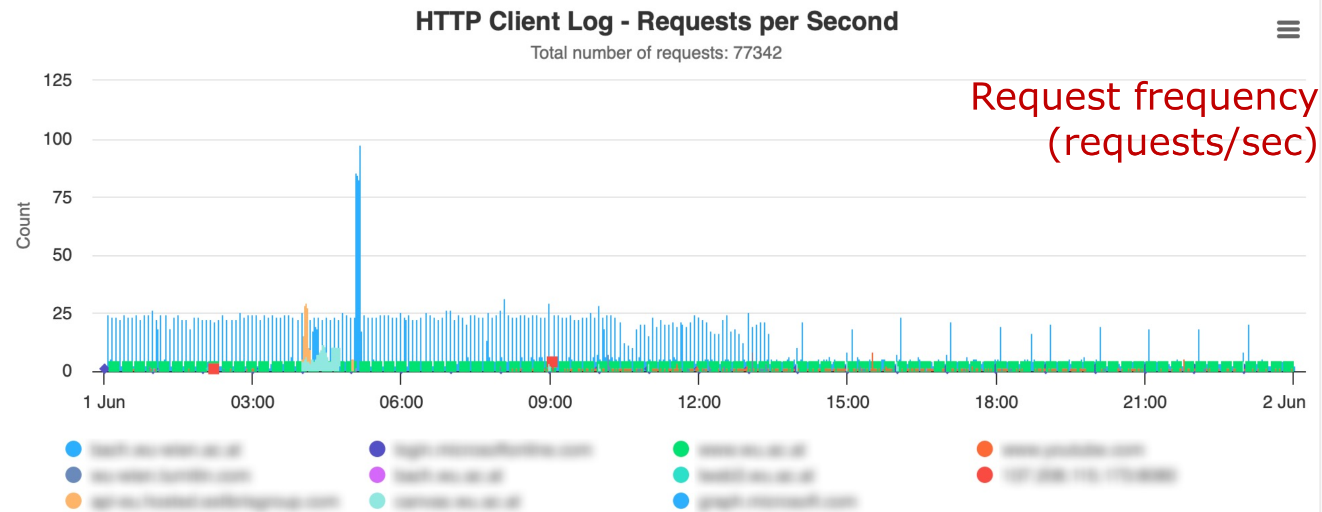
- What is ns_http:
  - Webserver performs as a web client requests from other servers
  - Cloud services, authentication, …
  - REST interfaces
  - Based on low-level server streaming infra-structure
  - Significantly faster than curl (esp. for high number of requests)
  - HTTP client request log (similar to access.log)
- What is new in NaviServer 5:
  - Persistent connections
  - Managing pool of connections, sharing across threads
- Challenges:
  - Requires strict error and parsing implementation (request pipelining)
  - Handling of streaming HTML (no content length provided)
  - Handling of incorrect replies
  - Handling of “100 continue”
  - …

ns_http reform (2/3)
Data visualized by NaviServer nsstats module

Often significant usage (up to several 100K client requests per day)



Here: bulk synchronization via ns_http with other systems mostly over night



ns_http reform (3/3)
Data visualized by NaviServer nsstats module

WIRTSCHAFTSUNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Statistics per server

Performance

Amount of Data

Status codes

Summative Statistics

Host	Requests	Avg Time	Sent	Received	200	201	202	204	400	403	404	408	429
[illegible]	5	742.46ms	706.82MB	11.06KB	4	1	0	0	0	0	0	0	0
[illegible]	3974	128.25ms	1.3MB	3.23MB	2532	0	0	0	7	0	0	0	1435
[illegible]	64246	122ms	19.86MB	294.07MB	64246	0	0	0	0	0	0	0	0
[illegible]	5	3.82s	1.46KB	3.87MB	5	0	0	0	0	0	0	0	0
[illegible]	8275	232.25ms	17.53MB	63.53MB	8274	0	0	0	0	0	0	1	0
[illegible]	3	189.84ms	5.37KB	2.2KB	1	0	0	1	0	0	1	0	0
[illegible]	24	176.96ms	9KB	58.04KB	24	0	0	0	0	0	0	0	0
[illegible]	1	594.84ms	375B	1.66KB	1	0	0	0	0	0	0	0	0
[illegible]	136	231.7ms	11.09MB	72.48MB	67	23	46	0	0	0	0	0	0
[illegible]	190	6.21ms	31.54KB	2.03MB	190	0	0	0	0	0	0	0	0
[illegible]	483	107.89ms	60.86KB	595.25KB	482	0	0	0	0	1	0	0	0

External servers often source of sudden performance bottlenecks

Removed occurrences of Double-Checking Lock Pattern

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

- Double-Checking Lock Pattern
  - Goal: reduce the overhead of acquiring locks
  - Testing the locking criterion before acquiring the lock.
- The Problem:
  - The pattern assumes a total store order (TSO), or the usage of “fences” (insert assembly)
  - In some language/hardware combinations, the pattern is unsafe (**RISC-V** has per default a weak memory order)
  - On x86: TSO, pattern is safe.

Newer architectures do aggressive optimizations, such as
1) compiler reordering instructions,
2) hardware reordering instructions,
3) cache coherency

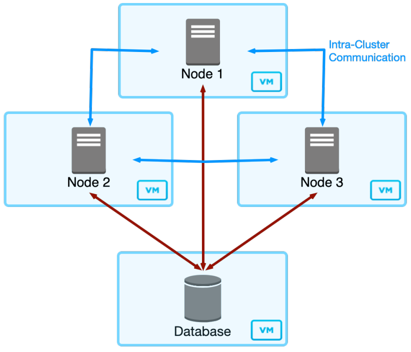
- NaviServer:
  - Two major variants of the double-checking lock pattern:
    1. start-up initialization
    2. lazy initialization of heap data (actually values kept for mutexes/locks, etc.)
  - Case 1: a posix/windows call can be used (pthread_once(), InitOnceExecuteOnce())
  - Case 2: requires more rewriting, lazy programming style.

```
/*
 * Core one-time server initialization to add a few Tcl_Obj
 * types.  These calls cannot be in NsTclInit above because
 * Tcl is not fully initialized at libnsd load time.
 */

if (!initialized) {
  Ns_MasterLock();
  if (!initialized) {
    Tcl_Obj *tmpObj = Tcl_NewIntObj(0);

    NS_intTypePtr = tmpObj->typePtr;
    Tcl_DecrRefCount(tmpObj);

    NsTclInitQueueType();
    NsTclInitAddrType();
    NsTclInitTimeType();
    NsTclInitMemUnitType();
    NsTclInitKeylistType();
    initialized = NS_TRUE;
  }
  Ns_MasterUnlock();
}
```

Large Scale NaviServer Configurations

- NaviServer provides detailed statistics, such as:
  - Mutex/RWLock statistics (see conference last year)
  - Requests (per connection pool)
  - Cache (requests, hits, flushes, savings, …)
  - Database (per DB pool, statements, performance, …)
  - …

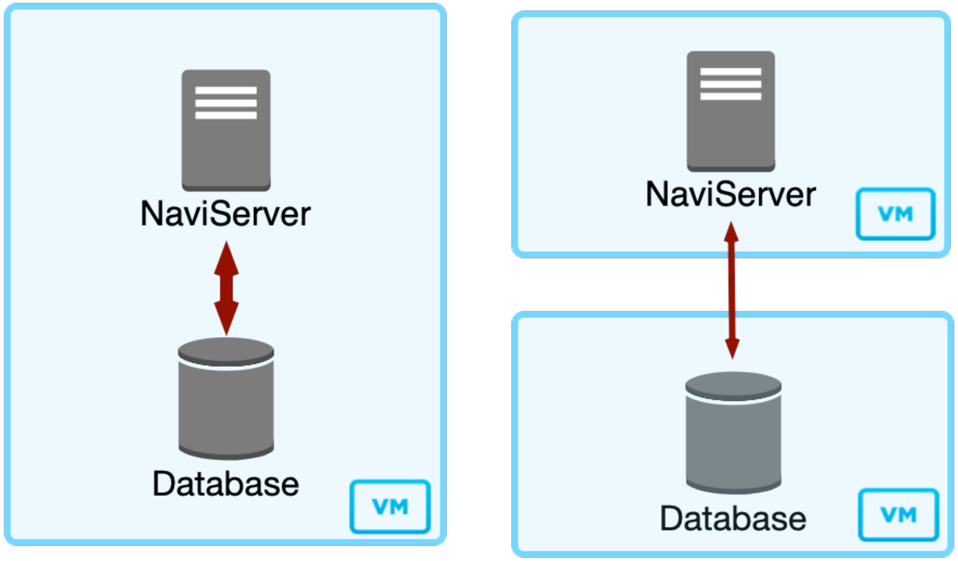
- OpenACS 5.10.1 has no cluster management:
  - Up to 5.10.1: static configuration, based on IP addresses
  - Not feasible for e.g. cloud operations
  - In 5.10.1: dynamic cluster configuration:
    - Additional cluster nodes can be registered/deregistered
    - Cluster join control via cluster secret



NaviServer Cluster with 3 Nodes, DB and NaviServers on different VMs

- Various trade-offs:
  - When DB and NaviServer are on the same machine
    - Communication with DB is fast
    - Maintaining cache coherency is relatively simple (all in one NaviServer instance)
    - NaviServer is excellent in making use of a high number of cores
  - But
    - What if this reaches limits?
    - Machines with many cores are still quite expensive
    - Can the throughput be doubled?
    - What are the consequences on response times (also on idle systems)?

Performance differences:
NaviServer and DB on the same or different VMs



Single NaviServer,
DB and NaviServer on the same VM

Single NaviServer,
DB and NaviServer on different VMs

- Common pattern: Database Server
- For cluster setups, DB is typically on an own VM
- Performance implications depend on application
  (e.g. how many SQL statements/request, cost of SQL requests)
- Network latency of assign 10 ms can cause throughput decrease by a factor of 20 based on pgbench, (see: https://www.cybertec-postgresql.com/en/postgresql-network-latency-does-make-a-big-difference/)

Empirical data from 3 sample OpenACS installations

	openacs.org	server1	server2
requests	448,029	7,642,282	4,308,318
response time/req (ms)	6.27	118.84	90.15
cache saving/req	5,3	69,45	113,72
cache flushes/req	0.0001	0.2406	1.3901
# SQL statements/req	3.93	38.66	22.33
SQL time/req	2.05	28.32	43.07

- Data collected when running servers over 4 days
- “server1” and “server2” are large sites, serving per day 1 mio requests or more
- Significant database use (server1: ~38 SQL statements per request, server2: ~22)
- Very few cache invalidations per request on OpenACS.org, very high on “server2”

Difference in response time and performance when running SQL server on a different VM

	openacs.org	server1	server2			
response time	6.27	118.84	90.15	1.00	1.00	1.00
remote SQL	8.32	147.16	133.22	1.33	1.24	1.48

	openacs.org	server1	server2			
max throughput (reqs/s	7022	505	666	1.00	1.00	1.00
remote SQL	5291	408	450	0.75	0.81	0.68

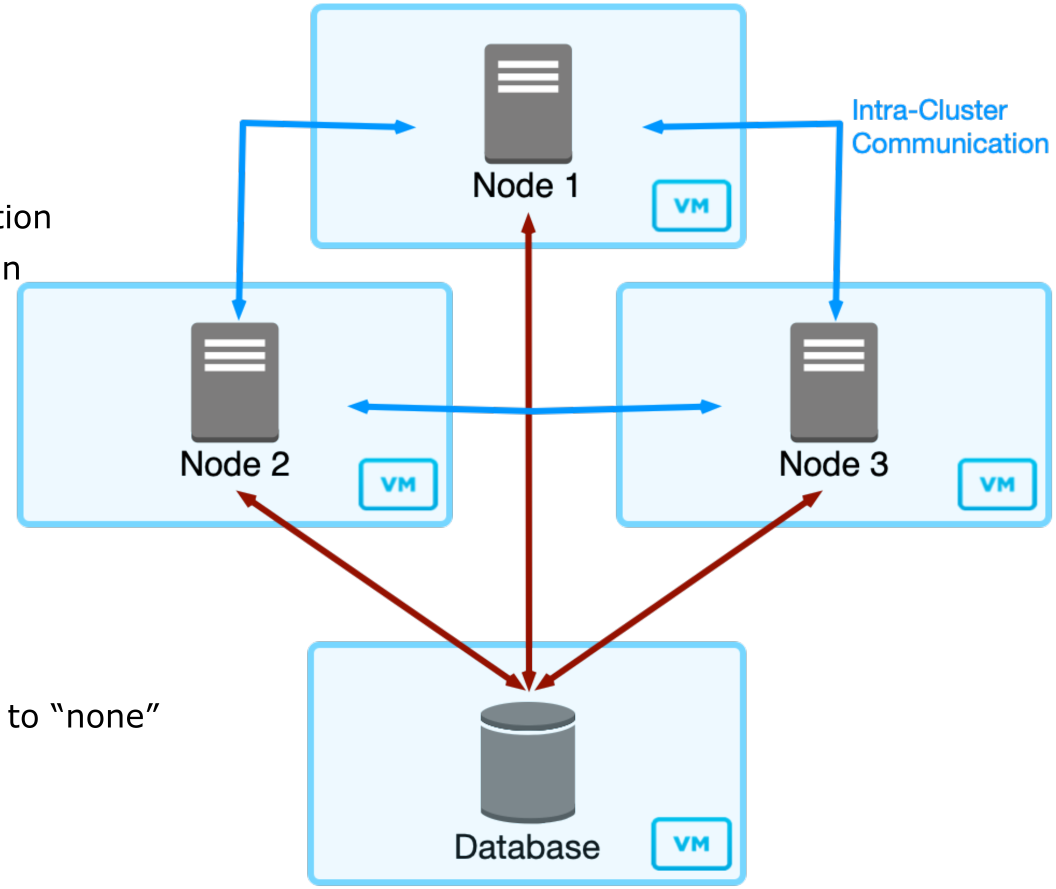
- Assumption:
  - remote SQL causes double latency per SQL statement (factor of 2)
  - For your applications: always best to measure, depends on local/cloud environment, etc.
- Average response time for openacs.org dropped by 30%, but still, it is fast enough, we are far from requiring max throughput.
- Drop of max throughput for ”server1” and “server2” might be sometimes already an issue, but probably, still OK

Performance differences:
NaviServer Cluster

- Example: 3 Nodes
- Database on a separate server

- For cache coherency:
  - Requires intra-cluster communication
  - Via HTTP/HTTPS/UDP/COAP built-in in NaviServer
  - Persistent connections handy and preferable
  - Requires updated applications, using "clusterwide" flush operations

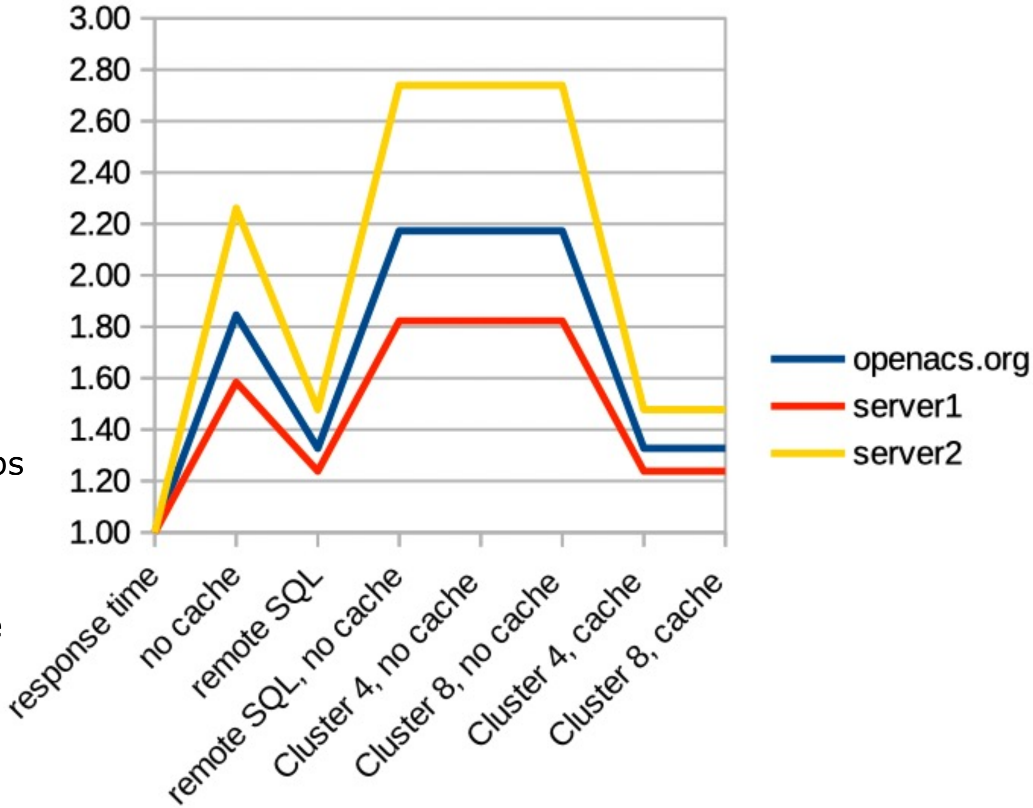
- Alternatively:
  - Avoid caching
  - Setting parameter "cachingmode" to "none"
  - Avoids most of intra-cluster communications with its overhead
  - But base performance degrades



NaviServer Cluster with 3 Nodes,
DB and NaviServers on different VMs

Performance implications for sample OpenACS installations

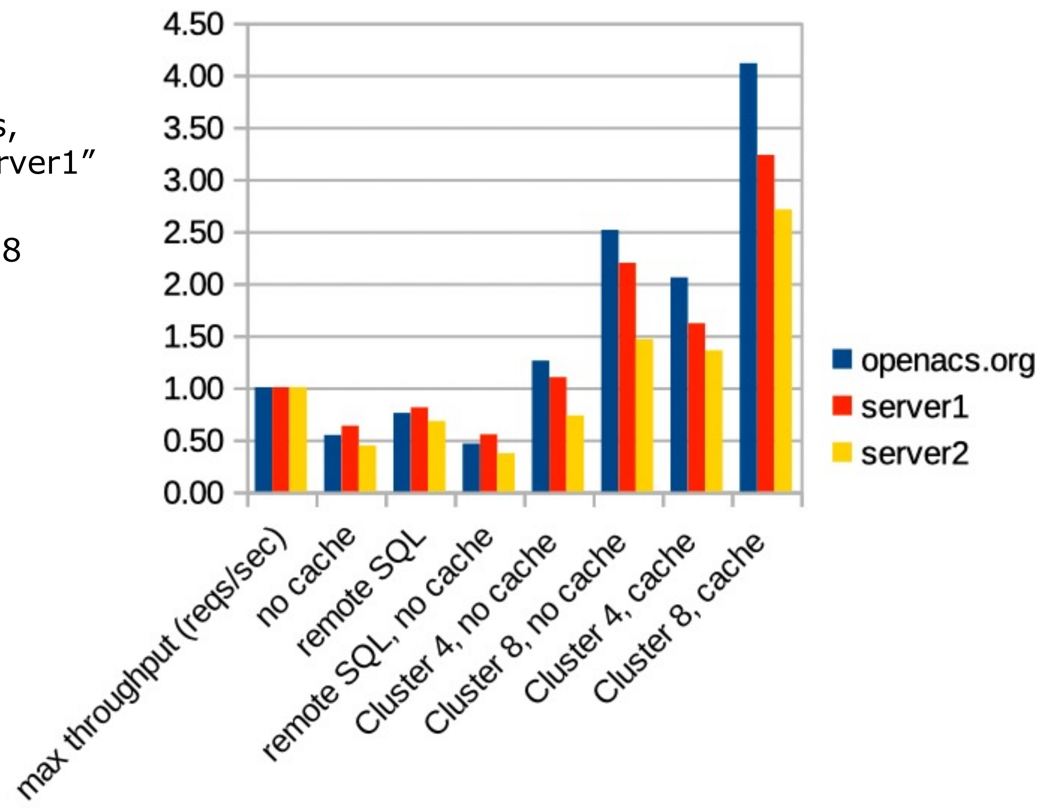
- Request Latency Comparison: comparing
  - Single server
  - Caching/no caching
  - Local SQL/remote SQL
  - Cluster nodes with 30 threads each
  - Cluster configuration with 4 nodes
  - Cluster configuration with 8 nodes
- Observation:
  - "server1" per-request performance drops most, when caching is deactivated (factor of 2.2)
  - Per-request performance of base configuration (DB + server on the same machine is best)
  - Caching benefits outweigh intra-cluster communication overhead



Throughput implications for sample OpenACS installations

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

- Throughput Observations:
  - With cluster "no cache" configurations, throughput of "openacs.org" and "server1" is already higher with 4 nodes.
  - Throughput can be doubled with 4 to 8 smaller cluster nodes
- Additional benefit:
  Higher availability in cluster configuration
- Caveats:
  - Is DB sufficiently scalable?
  - Statistics are collected from single VM installations

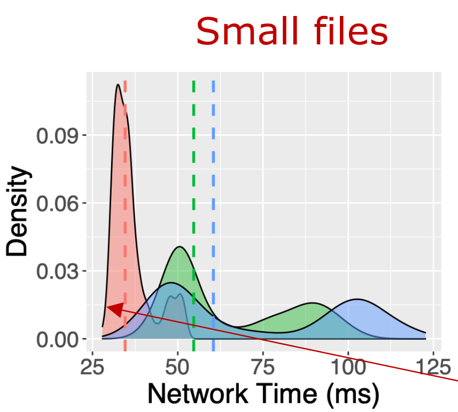


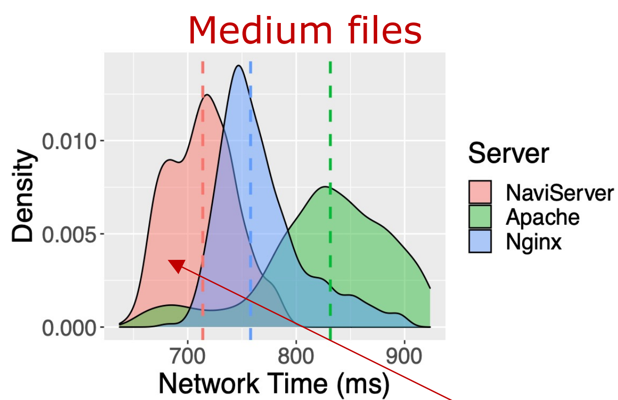
EQUIS ACCREDITED AACSB ACCREDITED AMBA ACCREDITED

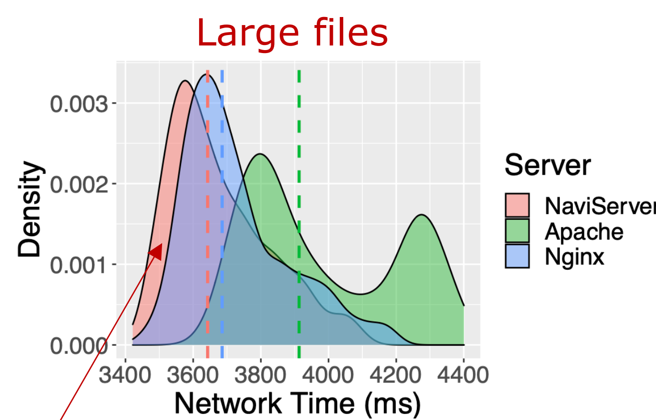
Experiment: HTTP/2 for NaviServer

Master Thesis of Philip Minić:

- Prototype version of NaviServer with HTTP/2 support
- Better performance than Apache and nginx with HTTP/2







NaviServer

Status

- Still experimental
- HTTP/3 (QUIC) is part of OpenSSL 3.1
- Still frequent changes in OpenSSL QUIC code base
- Little reason for HTTP/2 when HTTP/3 is available

Summary

- NaviServer 5
  - Overcomes many of the restrictions of NaviServer 4.99*
  - Strong integration with new Tcl 9 functionality
  - Many new features
- Learning from observation
  - Installations become more complex and distributed
  - Detailed monitoring eases
    - Configuration
    - Debugging
  - Still much to do!
  - Questions?

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Institute for Information Systems and New Media
Welthandelsplatz 1, 1020 Vienna, Austria

UNIV.PROF. DR. Gustaf Neumann

T +43-1-313 36-4671
Gustaf.neumann@wu.ac.at
www.wu.ac.at