TAUoverSupermon (ToS)

Low-Overhead Online Parallel Performance Monitoring

Aroon Nataraj, ***Matthew Sottile, Alan Morris,***

Allen D. Malony, Sameer Shende

{**anataraj**, matt, amorris, malony, shende}@cs.uoregon.edu

http://www.cs.uoregon.edu/research/tau

Department of Computer and Information Science

Performance Research Laboratory

Outline

- **Problem, Motivations & Requirements**
- **Our Approach - Coupling TAU and Supermon**
  - What is TAU? What is Supermon?
  - And how we coupled them?
- **Rationale**
- **Experimental Evaluation**
  - Online monitored data visualized
  - Performance / Scalability results investigated
  - Fault Tolerance demonstrated
  - Correlating System-level metrics with performance
- **Related Work**
- **Future Plans & Conclusion**

Performance Transport Substrate

- **Transport Substrate for Performance Measurement**
  - Enables communication with (and between) the performance measurement subsystems
  - Enables movement of measurement data and control
- **Modes of Performance Observation**
  - Offline / Post-mortem observation and analysis
    - least requirements for a specialized transport
  - Online observation
    - long running applications, especially at scale
  - Online observation with feedback into application
    - in addition, requires that the transport is bi-directional
- **Performance observation problems/requirements => Function of the mode => Addressed by substrate**

Performance Transport Substrate

- **Two fundamental components**
  - Measurement subsystem
  - Transport
- **Measurement Subsystem**
  - Instruments & measures application contexts (MPI ranks, processes, threads); Performance data producer
- **Transport**
  - Allows querying individual contexts' performance data
  - Bridges application (source) with monitors (sinks)
- **Challenge - low-overhead, scalable & fault-tolerant**
- **More problematic than post-mortem data retrieval**
  - Static: individual measurement contexts isolated
  - Online: global interactions between monitor and contexts

Primary Requirements of Transport Substrate

- **Performance of substrate**
  - Must be low-overhead
- **Robust and Fault-Tolerant**
  - Must detect and repair failures
  - Must not adversely affect the application on failures
- **Bi-directional Transport (Control)**
  - Selection of events, measurement technique, target nodes
  - What data to output, how often and in what form?
  - Feedback into the measurement system & application
  - Allows synchronization between sources & sinks
- **Scalable**
  - Must maintain the above properties at scale

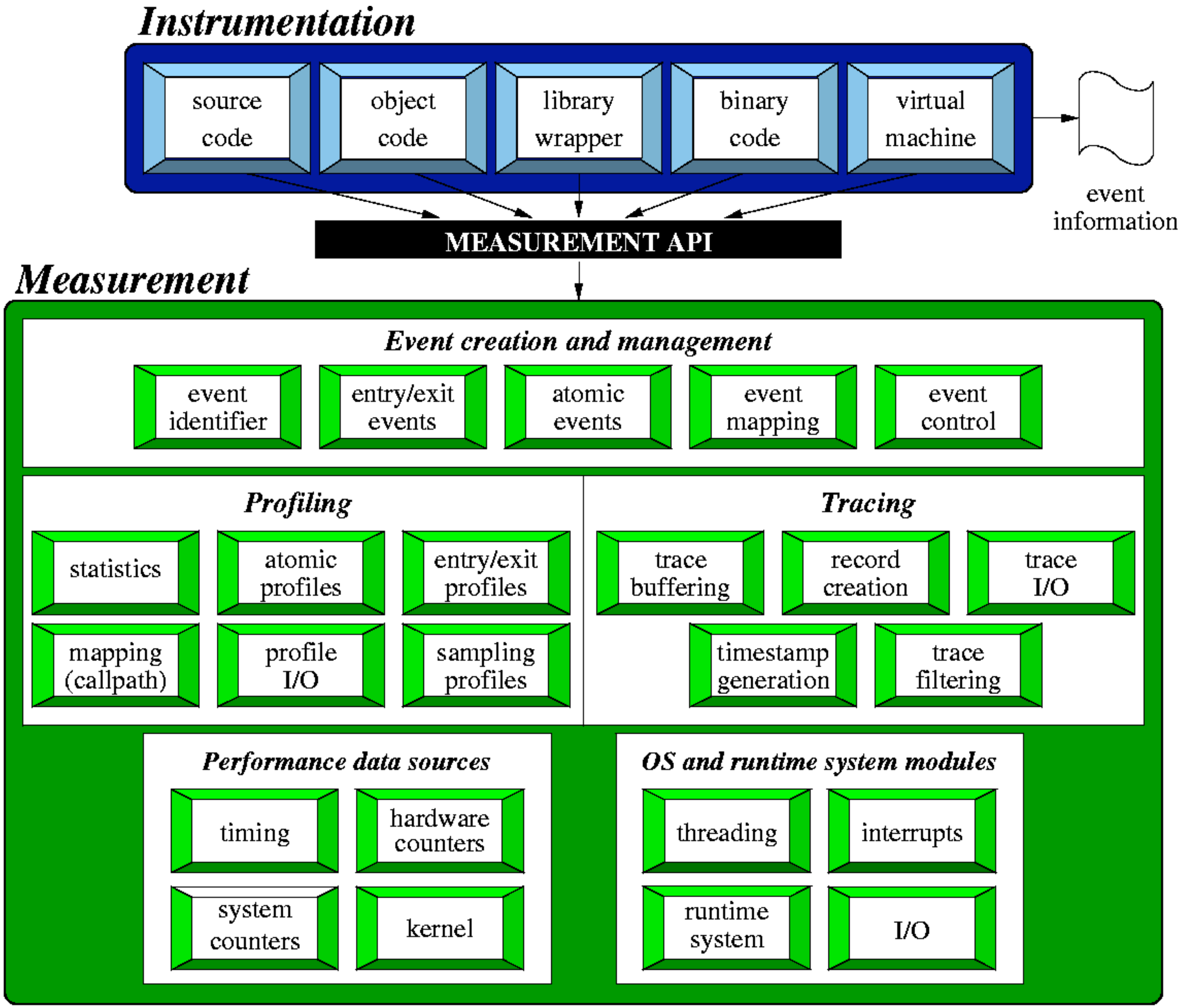
Secondary Requirements

- **Data Reduction**
  - At scale, cost of moving data too high
  - Allow sampling/aggregation in different domains (node-wise, event-wise)
- **Online, Distributed Processing of generated performance data**
  - Use compute resource of transport nodes
  - Global performance analyses within the topology
  - Distribute statistical analyses
    - easy (mean, variance, histogram), challenging (clustering)

What is TAU?

- ***T***uning and ***A***nalysis ***U***tilities (14+ year project effort)
- **Performance system framework for HPC systems**
  - Integrated, scalable, flexible, and parallel
  - Multiple parallel programming paradigms
  - Parallel performance mapping methodology
- **Portable (open source) parallel performance system**
  - Instrumentation, measurement, analysis, and visualization
  - Portable performance profiling and tracing facility
  - Performance data management and data mining
- **Scalable (very large) parallel performance analysis**
- ***Partners***
  - Research Center Jülich, LLNL, ANL, LANL, UTK

TAU Performance System Architecture



TAU Measurement Mechanisms

- **Parallel profiling**
  - Function-level, block-level, statement-level
  - Supports user-defined events and mapping events
  - TAU parallel profile stored (dumped) during execution
  - Support for flat, callgraph/callpath, phase profiling
  - Support for memory profiling (headroom, leaks)
- **Tracing**
  - All profile-level events
  - Inter-process communication events
  - Inclusion of multiple counter data in traced events
- Compile-time and runtime measurement selection

What is Supermon?

- **Scalable monitoring system for HPC clusters**
  - From LANL [12] (Sottile & Minnich)
  - Sockets-based servers
  - Gather, organize and transfer monitoring data
- **Heirarchical Architecture**
  - Mon-Daemons as leaves
  - Root and Intermediate Supermon daemons
- **Data format - Symbolic expressions**
  - s-exprs based on LISP - unit of transport (like a packet)
- **Primary purpose:** Monitoring system-level performance reported by H/W sensors and OS performance data.
- **In addition, listens on a UDS port, called *monhole***
  - Allows custom data to be sent over the Supermon channels

Approaches

- **Option 1: Use a NFS from within TAU and monitors**
  - Global shared-FS must be available
  - File I/O overheads can be high
  - Control through file-ops costly (e.g. meta-data ops)
  - All data not consumed - persistent storage wasteful
- **Option 2: Build new custom, light-weight transport**
  - Allows tailoring to TAU
  - Significant programming investment
  - Portability concerns across
  - Re-inventing the wheel, may be
- **Our approach: Re-use existing transports**
  - Transport plug-ins couple with and adapt to TAU

Approaches continued …

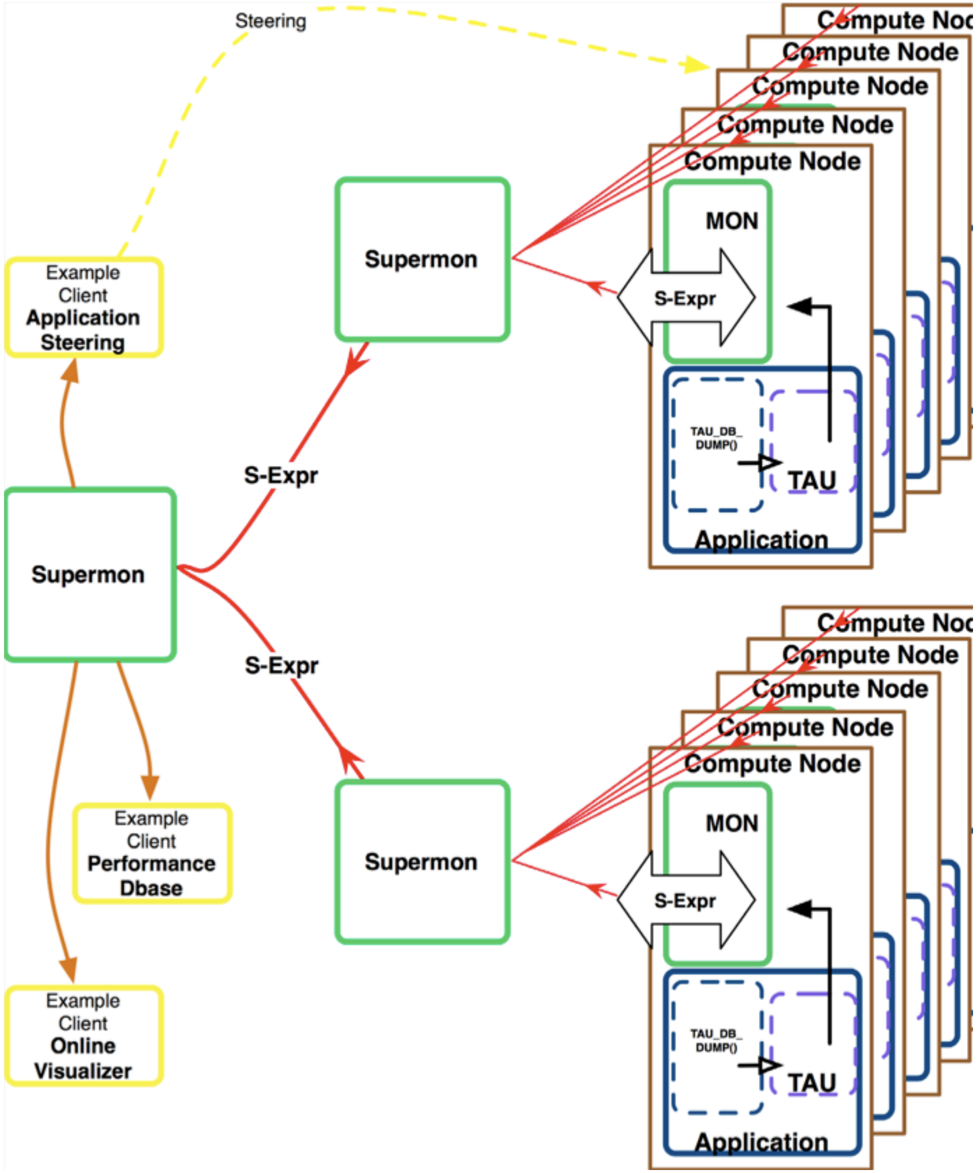
- **Measurement and data transport separated**
  - No such distinction in TAU before
- **Created abstraction to separate and hide transport**
  - *TauOutput*
- **TauOutput exposes subset of Posix File I/O API**
  - Acts as a *virtual transport layer*
  - Most of TAU code-base unaware of transport details
  - Very few changes in code-base outside adapter
- **Supermon (Sottile and Minnich, LANL) Adapter**
  - TAU instruments & measures
  - Supermon bridges monitors (sinks) to contexts (sources)

Rationale

- **Moved away from NFS**
- **Separation of concerns**
  - Scalability, portability, robustness, fault-tolerance
  - Addressed independent of TAU
- **Re-use existing technologies where appropriate**
- **Multiple bindings**
  - Use different solutions best suited to particular platform
- **Implementation speed**
  - Easy, fast to create adapter that binds to existing transport
- **Performance correlation : Bonus**

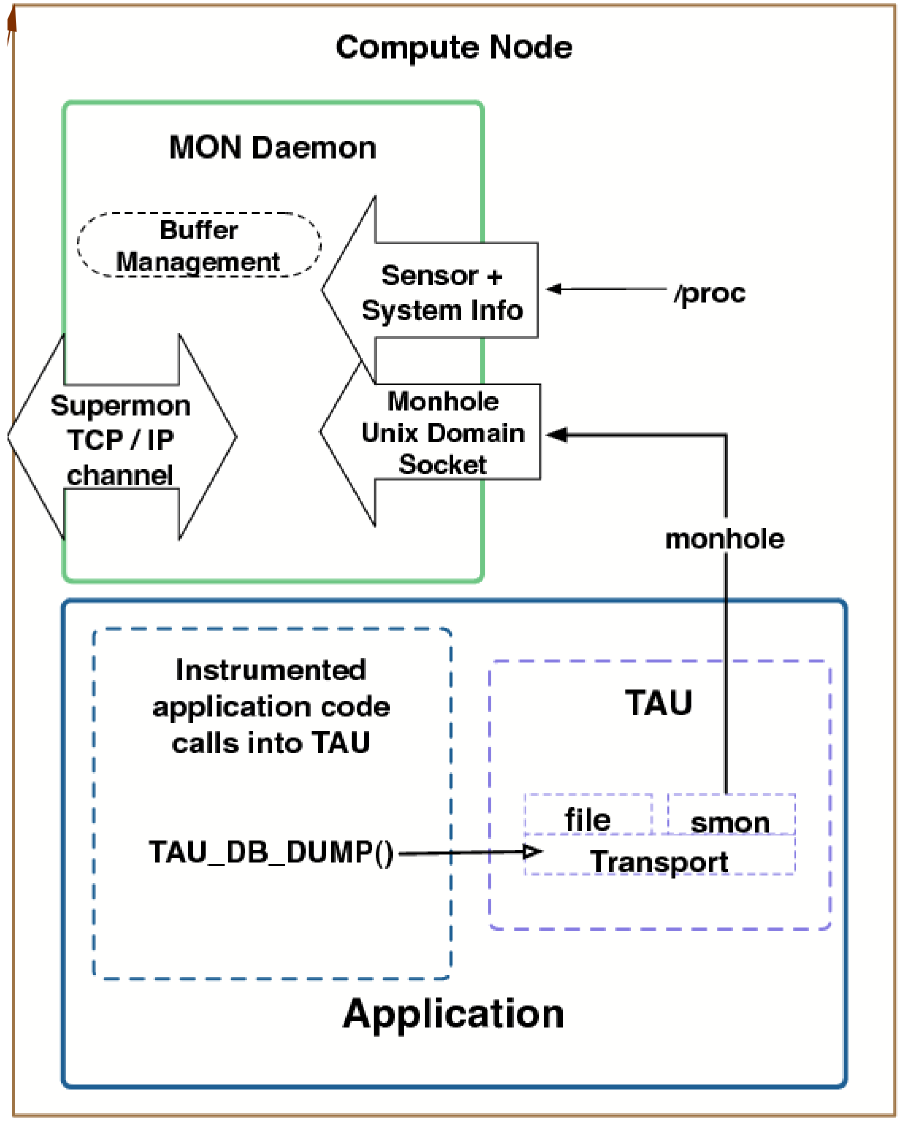
ToS Architecture

- **TAU**
  - Front-End (Sink)
  - ToS-Adapter
  - Back-End Application
- **Supermon**
  - Root & Internal Supermon-D
  - MON-D (Compute Node)
- **Data Retrieval**
  - Push-Pull Model
- **Multiple Sinks**

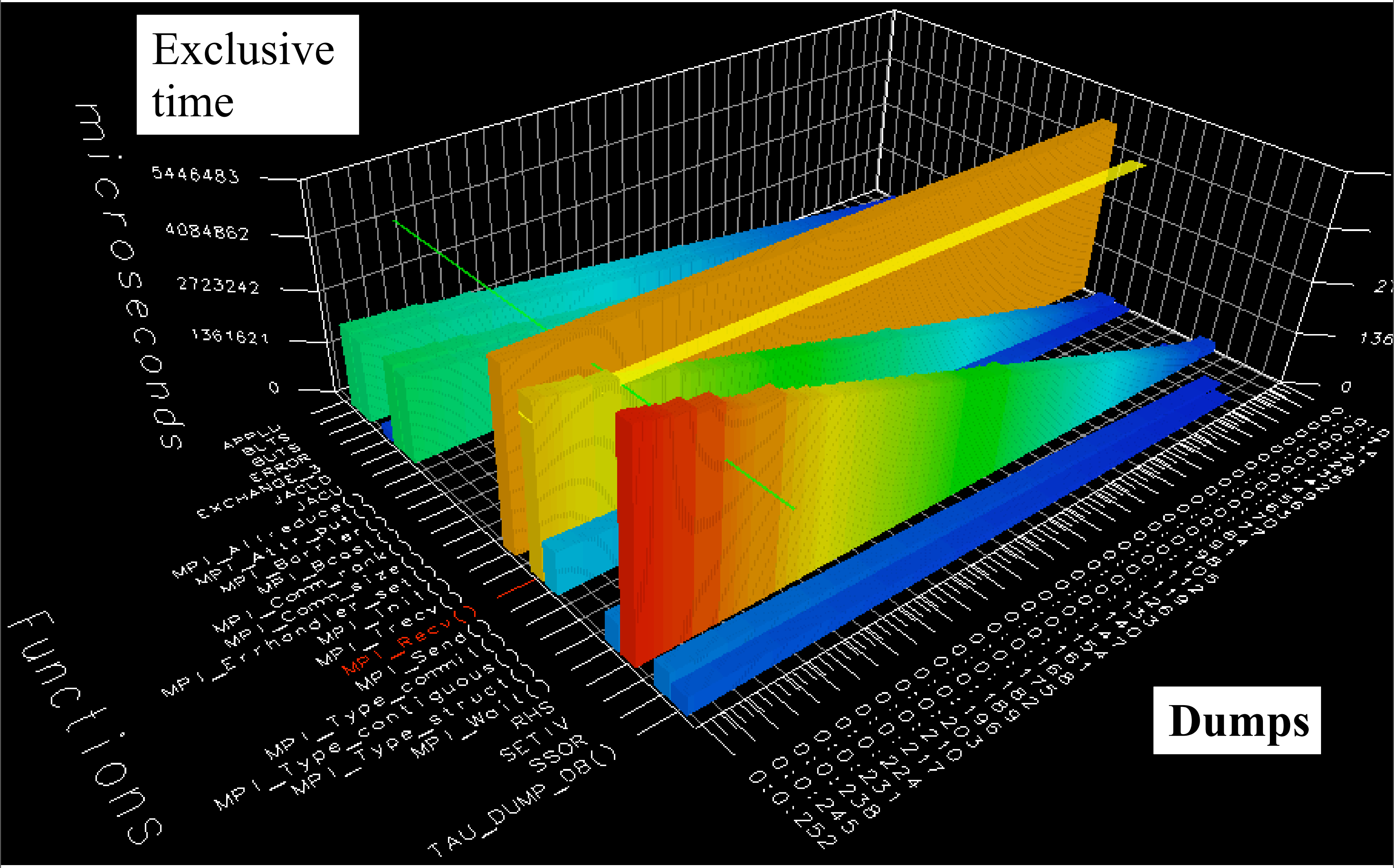


ToS Architecture - Back End

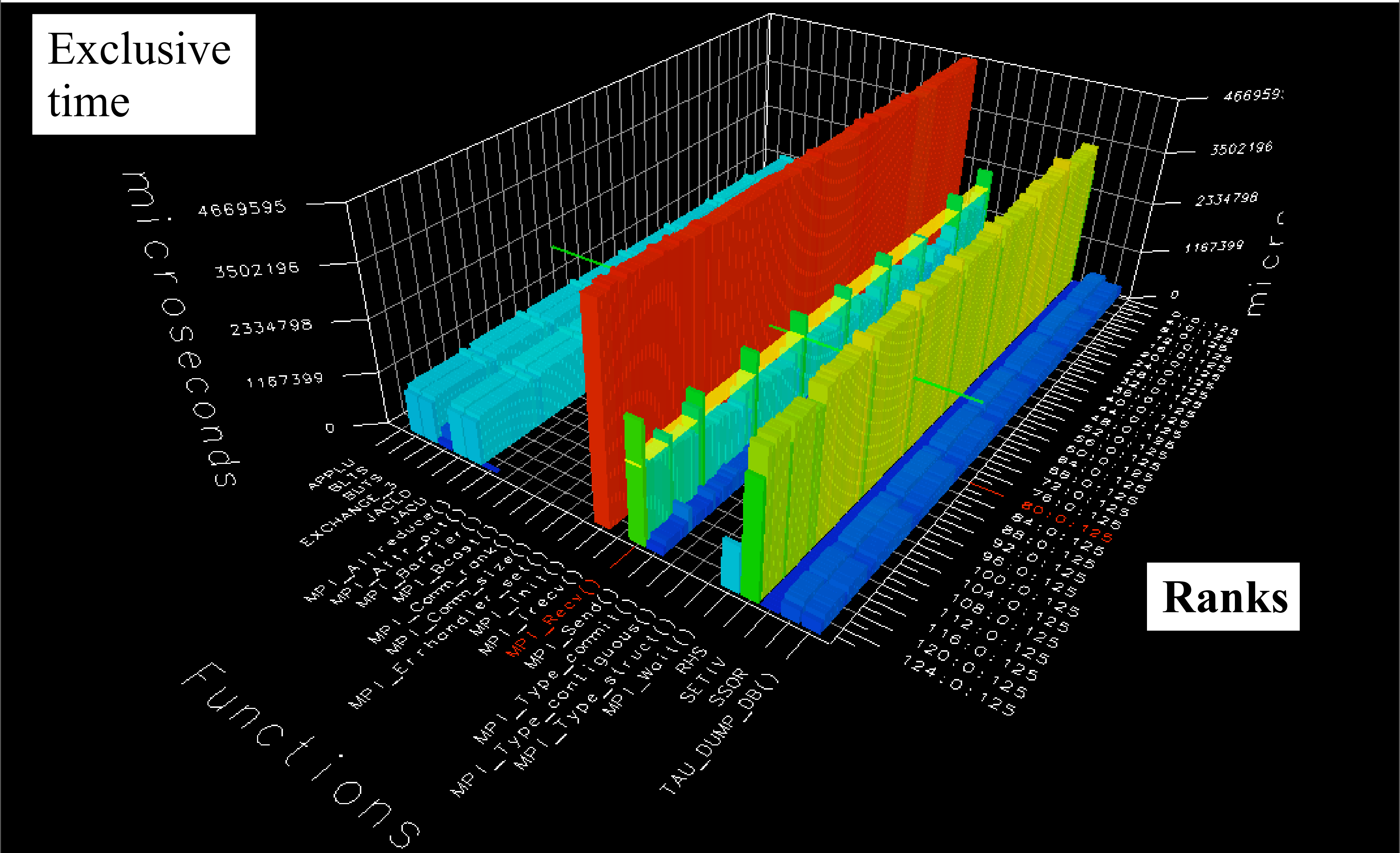
- **Application calls TAU**
  - Per-Iteration call to *dump*
  - Periodic calls using *alarm*
- **TauOutput object**
  - Configuration specific: compile or runtime
  - One per thread
  - Exposes subset of Posix File-I/O operations
- **Non-blocking *recv Ctrl***
- **Source push; Sink pull**
- **System, HW Sensor**



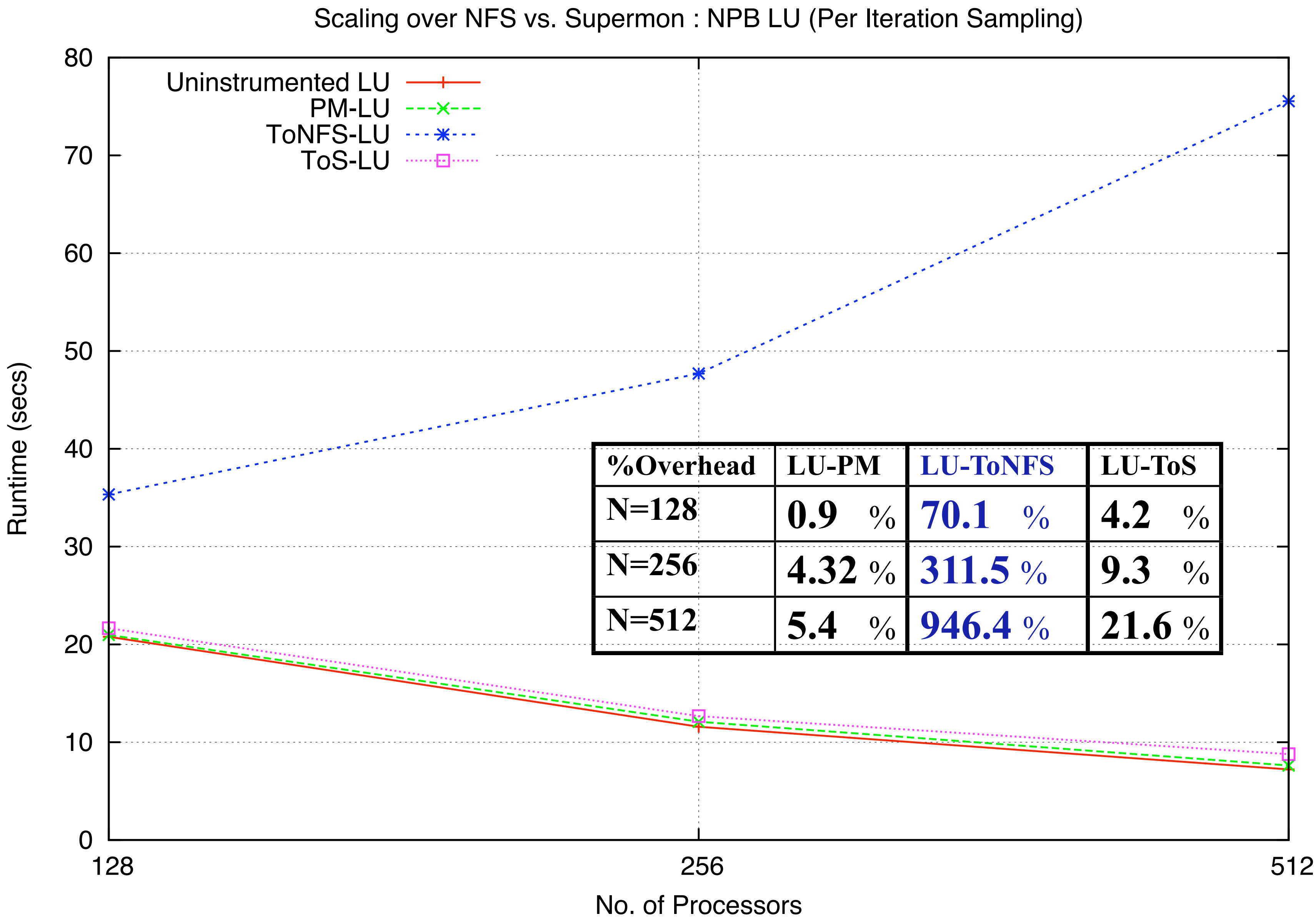
Simple Example (NPB LU-C) Rank 0 Dump-View



Simple Example (NPB LU-C) 1 Dump Rank-View



Performance & Scalability



%Overhead	LU-PM	LU-ToNFS	LU-ToS
N=128	0.9 %	70.1 %	4.2 %
N=256	4.32 %	311.5 %	9.3 %
N=512	5.4 %	946.4 %	21.6 %

Performance & Scalability

- **Observations**
  - TAU measurements (LU-PM) contributed *atmost* **5.4%** overhead (and as low as **0.9%** for N=128).
  - Online measurement & data retrieval using NFS leads to *atleast* **70%** and *grows super-linearly* to over **946%**.
  - Online measurement & data retrieval using ToS is (2) orders of magnitude better than ToNFS.
  - ToS as low as 4.2%, but does grow to 22%.
  - Severe stress test. 253 dumps in approx. 7 seconds, each from 512 processes. For typical monitoring will be small.
  - As LU scales (strongly), *savings* from using ToS over ToNFS also *super-linear*.

Performance & Scalability - Digging Deeper

- **Examine difference between LU-PM & LU-ToNFS**
- **The DUMP itself does not explain the gap**
- **Where did the time go?**

(in seconds)	**NFS - PM**	**NFS Dump**	**Gap?**
N=128	14.36	7.428	6.932
N=256	35.6	15.834	19.765
N=512	67.94	32.367	35.573

Performance & Scalability - Digging Deeper

- **Examine difference between LU-PM & LU-ToNFS**
- **The DUMP itself does not explain the gap**
- **Where did the time go?**

(in seconds)	**NFS - PM**	**NFS Dump**	**Gap?**	**Mpi_Recv Diff**
N=128	14.36	7.428	6.932	7.023
N=256	35.6	15.834	19.765	19.886
N=512	67.94	32.367	35.573	35.745

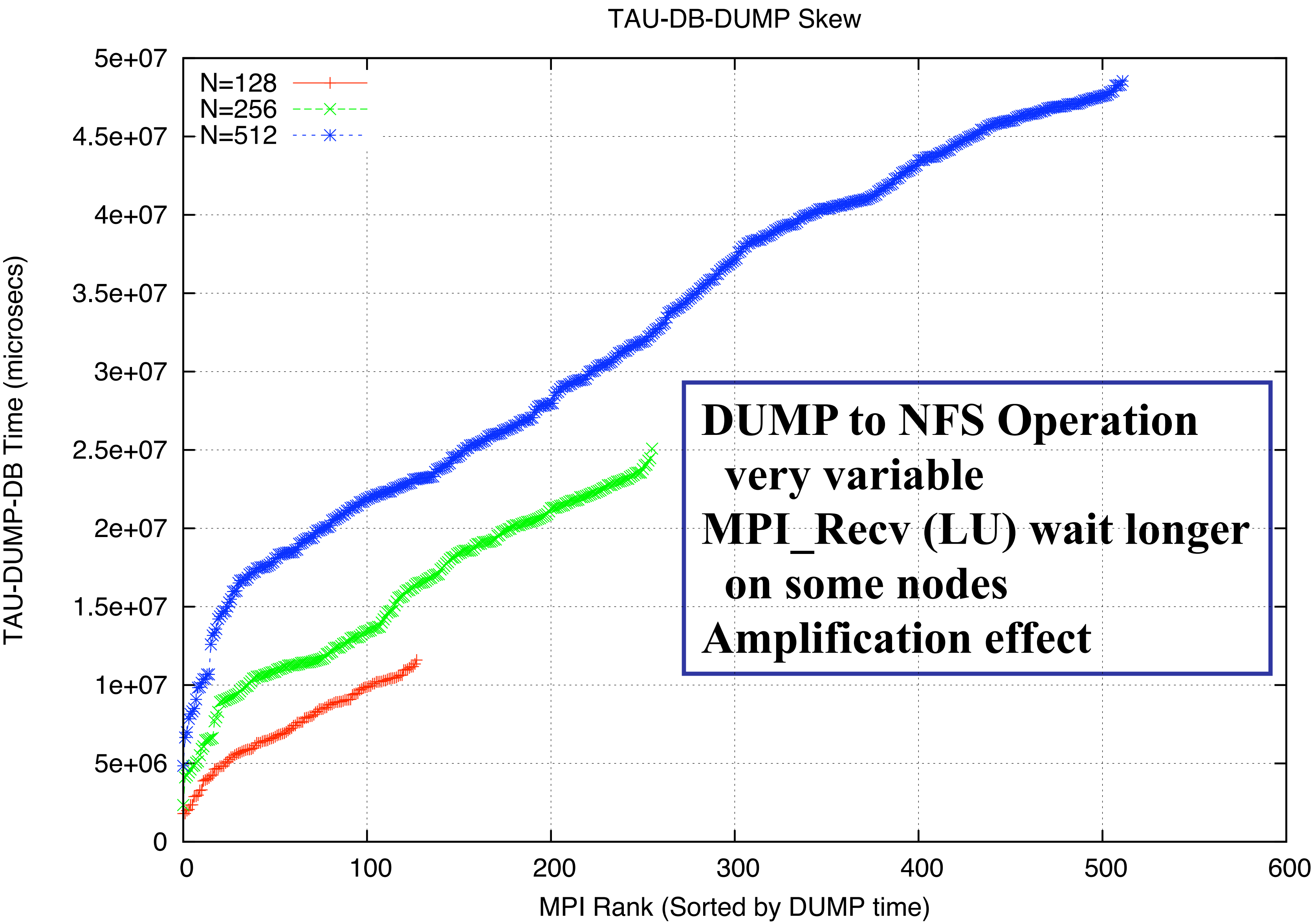
Performance & Scalability - Digging Deeper

- **Examine difference between LU-PM & LU-ToNFS**
- **The DUMP itself does not explain the gap**
- **Where did the time go?**

(in seconds)	**NFS - PM**	**NFS Dump**	**Gap?**	**Mpi_Recv Diff**
N=128	14.36	7.428	6.932	7.023
N=256	35.6	15.834	19.765	19.886
N=512	67.94	32.367	35.573	35.745

- **Longer MPI_Recv in ToNFS explains the gap…**
- **Why?**

Performance & Scalability - Digging Deeper



DUMP to NFS Operation very variable
MPI_Recv (LU) wait longer on some nodes
Amplification effect

Performance & Scalability - Digging Deeper

- **Kernel-level profile (from KTAU)**
  - Smaller run (4 ranks) on test cluster at UO
  - System call durations (in seconds)

Type	rename	select	open	writev	read	close	write
Tau-NFS	**11.75**	9.46	**8.55**	4.02	3.22	2.50	0.63
Tau-PM	**0**	5.94	**0.03**	3.95	3.22	0	0.60

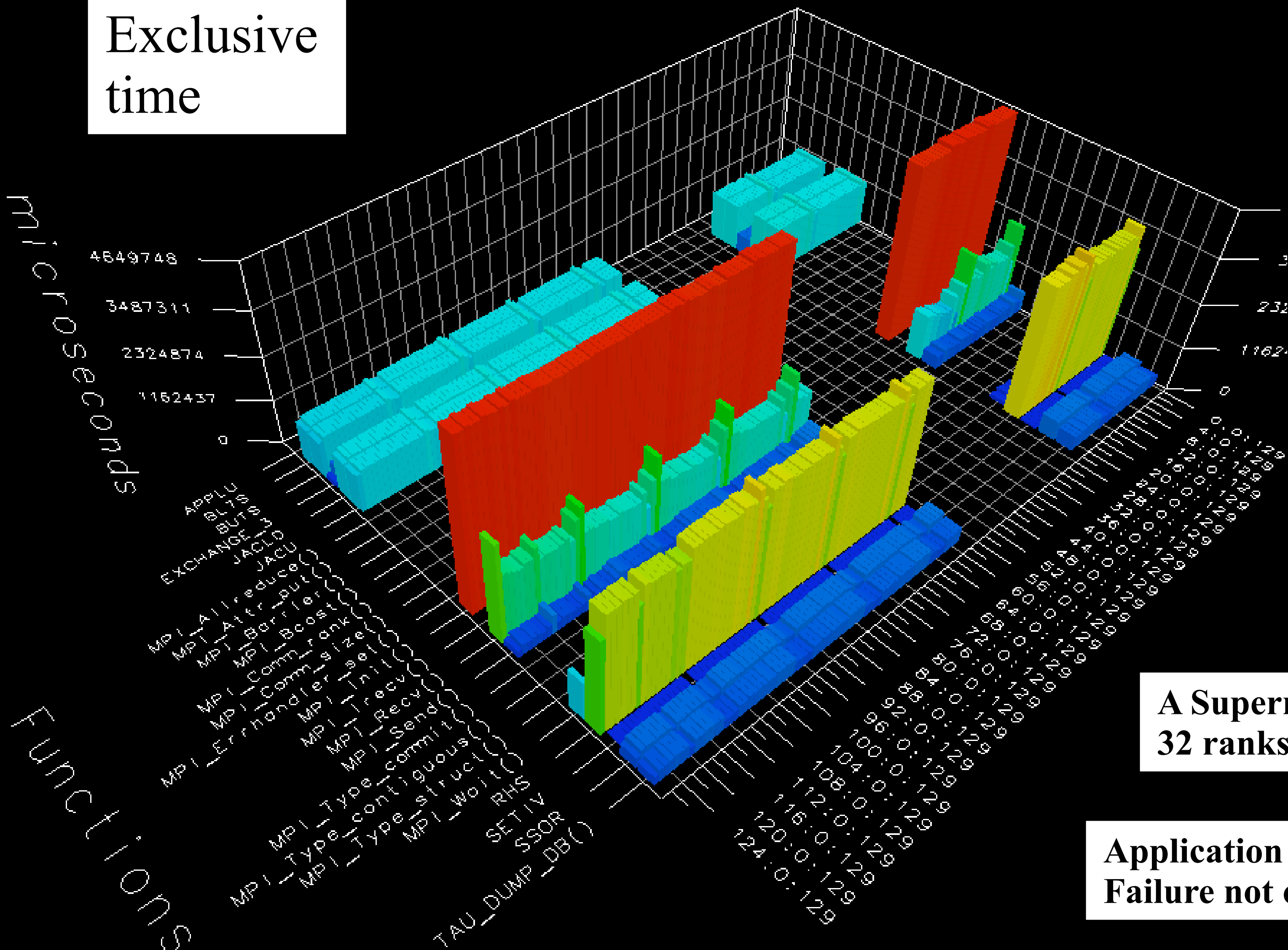
- ***sys_open()***, ***sys_renamc()*** dominate cost
  - Lack of proper control forces use of meta-data ops
  - Used to synchronize between producers & consumers
- ***sys_select()*** shows the extra waiting in MPI_Recv
- **NFS Dump operation costs**
  - Meta-Data operations
  - Blocking & Variable

Fault Tolerance Demonstrated

- **Goals**
  - Insulate Application
  - Recover from Failure
  - Isolate failures to sub-tree
- **Crash Failure - Supermon crashes**
  - kill() intermediate Supermon
  - Sub-tree performance data lost
  - Other Ranks' data still available
- **Conn Failure - Connection broken**
  - Perform a Crash, then restart Supermon
  - Tree gets repaired automatically
- **Application Successful - No slowdown**

Fault Tolerance - Scenario: Supermon Crash

Exclusive time

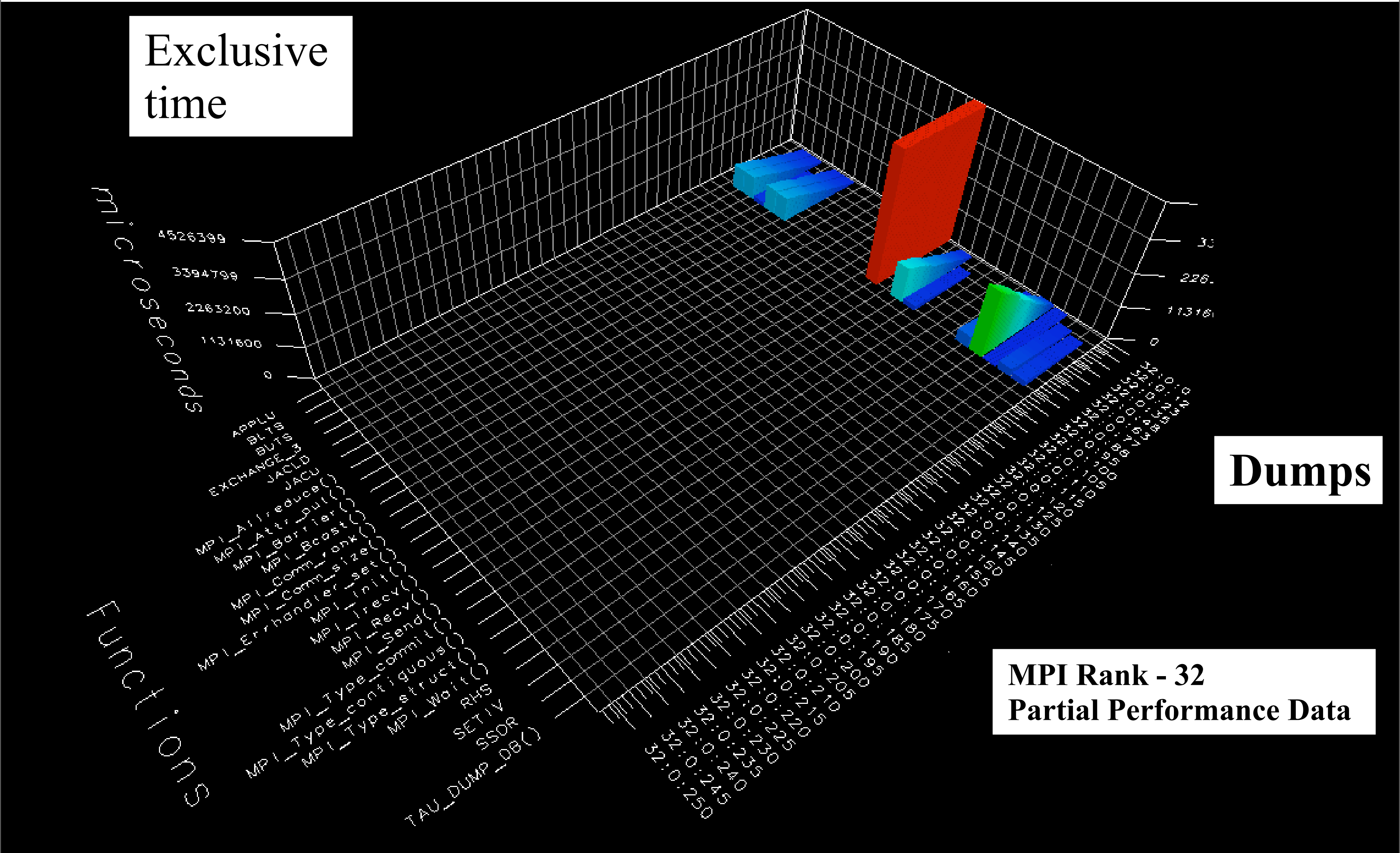


Ranks

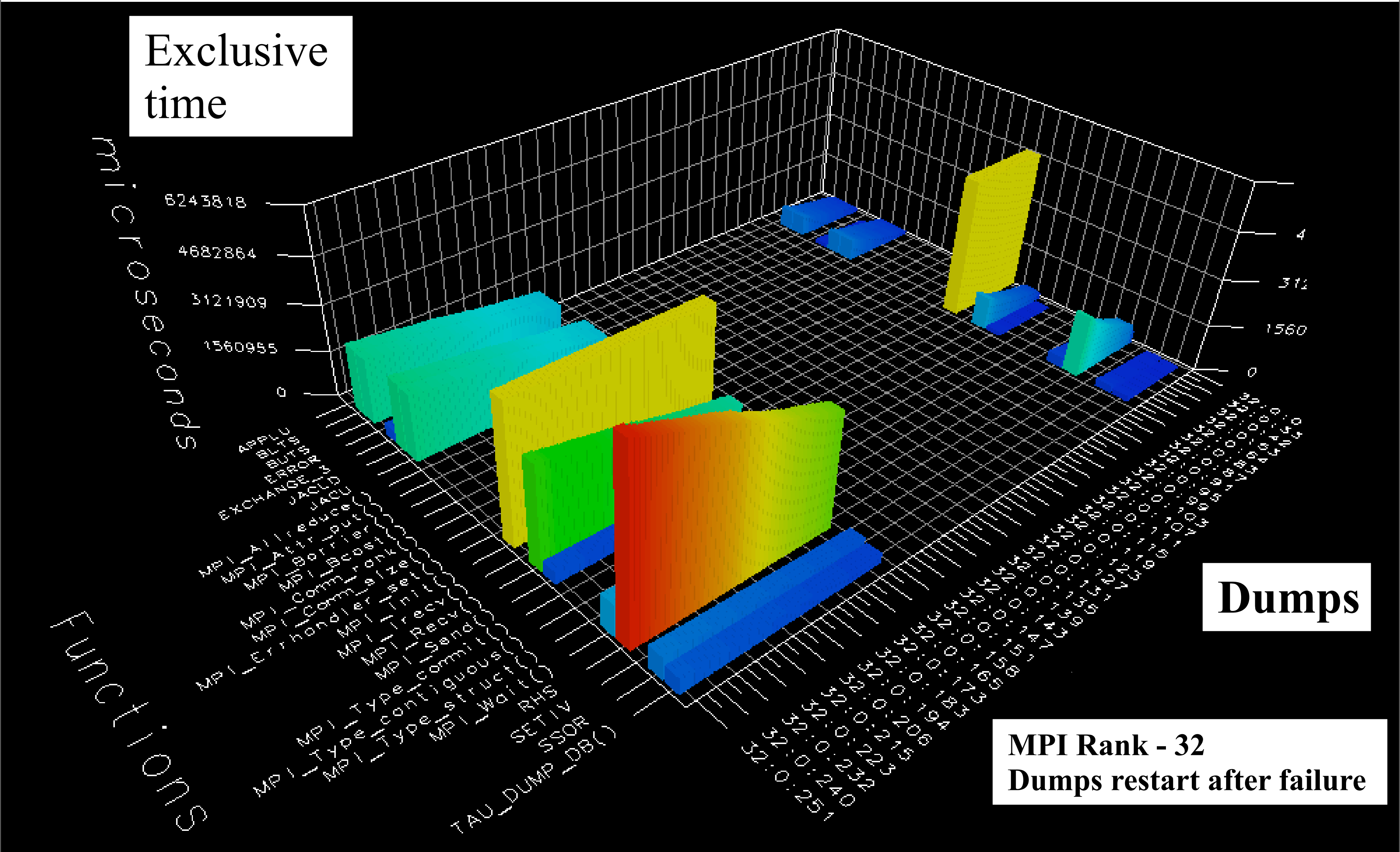
A Supermon crashes. 32 ranks no data.

Application successful Failure not catastrophic.

Fault Tolerance - Scenario: Supermon Crash



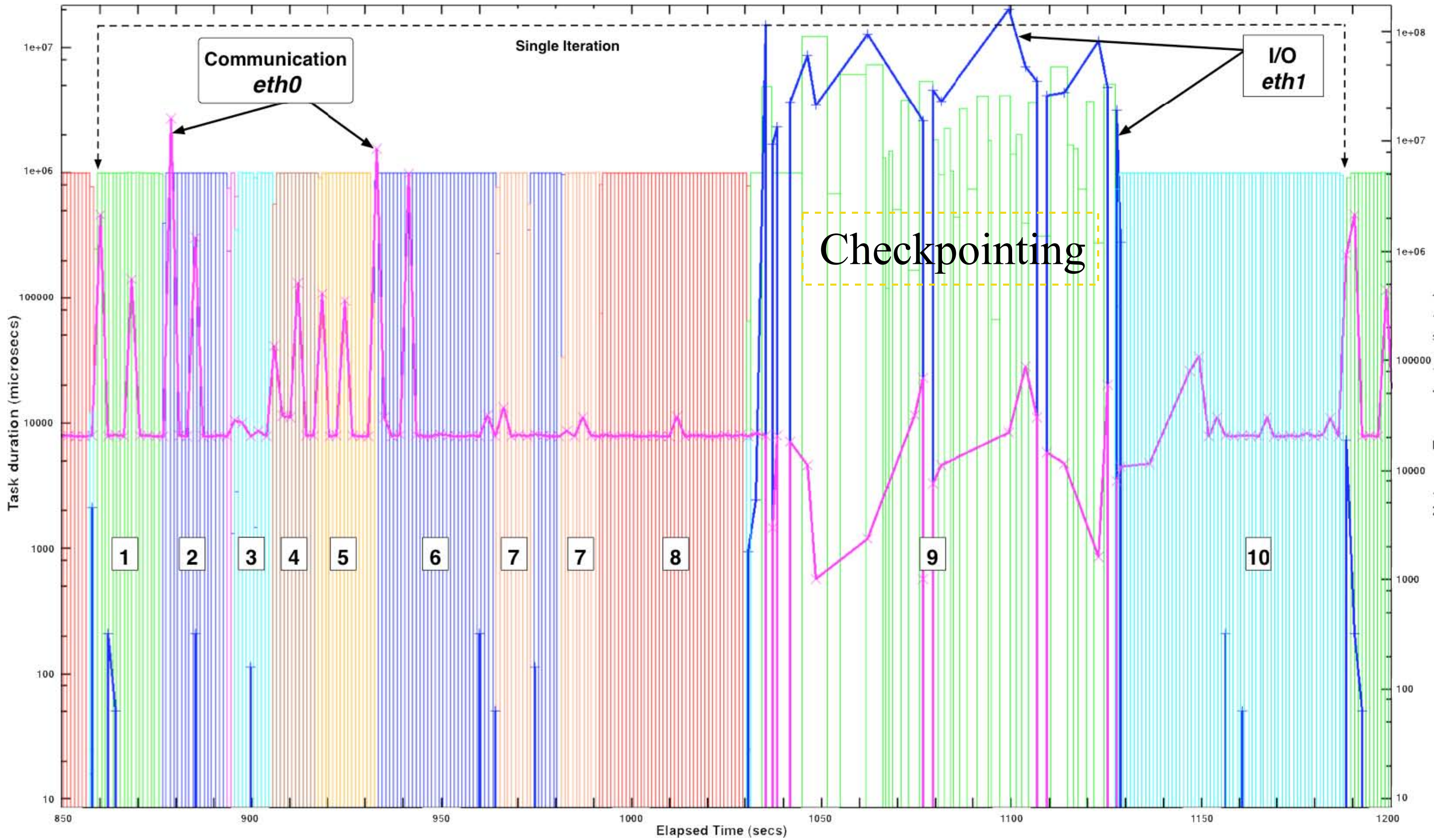
Fault Tolerance - Scenario: Connection Drop



Fault Tolerance Explained

- **Supermon is loosely coupled**
  - Failures do not propagate back
  - Keeps a list of dead-connections
- **Front-End periodically sends out recursive-repair command**
- **Back-End connection failure not catastrophic**
- **Summary**
  - Connection Failures are handled transparently
  - Crash Failures not catastrophic (s/w crash may be handled transparently with watch-dog)
- **Limitations**
  - Only Fail-Stop errors examined
  - Does not route-around crash failures

Correlation with System Metrics (Uintah)



Uintah phase data from TAU; Network Metrics from Supermon

Related Work

- Online automated computational ***steering frameworks*** - Falcon [4], Autopilot [1], Active Harmony [2] and MOSS [6]; sensors, actuators; ***built-in transport***.
- Distributed Performance Consultant [3], online ***performance diagnosis*** - uses high-performance data transport, reduction system, MRNET [7].
- OMIS [10] and OCM [9] - universal ***interface*** between tools and monitoring system. Event-action paradigm; distributed client-server system for monitoring services.
- Periscope [11] - ***hierarchical monitoring agents*** (reduction & performance evaluation) - ***scalable***.
- ToS, ***couples*** two independent, mature, stand-alone systems; builds on lower-level interface between TAU & Supermon; underlying ***virtual-transport-layer*** allows other ***adaptors***; other adaptors feasible (TAUoverMRNET in progress); steering not applied - but ***reverse control channel*** exists.

Conclusion & Future Work

- **ToS: Online parallel performance monitor that is *low-overhead*, *scalable* and *fault-tolerant*, based on coupling TAU with Supermon**
- **Further Experimentation**
  - More complex applications, workloads
  - Different topologies & Buffer strategies (see paper)
- **Improving ToS**
  - Add ability to perform Data-Transformations
  - Improve Control capabilities
- **Adapters: MRNET (Arnold, Miller) adapter (WIP).**
- **Platforms: ToS BG/L port. I/O Node hosts Mon-D.**
  - Need to tune and experiment

Acknowledgements

- Matt Sottile, Ron Minnich (Los Alamos National Lab) for Supermon collaboration and guidance
- Computing Resource Access
  - Argonne National Lab for access to the BG/L system
  - Lawrence Livermore National Lab for Linux cluster access provided to the TAU group
- Funding for ToS
  - Los Alamos National Laboratory

References

- [1] : Tapus, C., Chung, I.H., Hollingworth, J.: Active harmony: Towards automated performance tuning. In: SC'02: ACM/IEEE conference on Supercomputing (2002).
- [2] : Ribler, R., Simitci, H., Reed, D.: The Autopilot performance-directed adaptive control system. Future Generation Computer Systems 18(1), 175–187 (2001).
- [3] : Miller, B., Callaghan, M., Cargille, J., Hollingsworth, J., Irvin, R., Karavanic, K., Kunchithapadam, K.,Newhall, T.: The paradyn parallel performance measurement tool. Computer 28(11), 37–46 (1995).
- [4] : Gu, W., et al.: Falcon: On-line monitoring and steering of large-scale parallel programs. In: 5th Symposium of the Frontiers of Massively Parallel Computing, McLean, VA, pp. 422–429 (1995).
- [5] : Nataraj, A., Malony, A., Shende, S., Morris, A.: Kernel-Level Measurement for Integrated Parallel Performance Views: the KTAU Pro ject. In: CLUSTER'06: International Conference on Cluster Computing, Barcelona, Spain (2006).
- [6] : Eisenhauer, G., Schwan, K.: An ob ject-based infrastructure for program monitoring and steering. In: 2nd SIGMETRICS Symposium on Parallel and Distributed Tools (SPDT'98), pp. 10–20 (1998).
- [7] : Roth, P., Arnold, D., Miller, B.: Mrnet: A software-based multicast/reduction network for scalable tools. In: SC'03: ACM/ IEEE conference on Supercomputing (2003).
- [8] : Bailey, D.H., et al.: The nas parallel benchmarks. The International Journal of Supercomputer Applications 5(3), 63–73 (1991).
- [9] : Wismuller, R., Trinitis, J., Ludwig, T.: Ocm – a monitoring system for interoperable tools. In: 2nd SIGMETRICS Symposium on Parallel and Distributed Tools (SPDT'98), pp. 1–9 (1998).
- [10] : Ludwig, T., Wism ¨uller, R., Sunderam, V., Bode, A.: Omis – on-line monitoring interface specification (version 2.0). LRR-TUM Research Report Series 9 (1998).
- [11] : Gerndt, M., F ¨urlinger, K., Kereku, E.: Periscope: Advanced techniques for performance analysis. In: In: Parallel Computing: Current & Future Issues of High-End Computing, In the International Conference ParCo (2005).
- [12] : Sottile, M., Minnich, R.: Supermon: A high-speed cluster monitoring system. In: CLUSTER'02: International Conference on Cluster Computing (2002).
- [13] : Shende, S., Malony, A.D.: The TAU parallel performance system. The International Journal of High Performance Computing Applications 20(2), 287–331 (2006).
- [14] : Huck, K.A., Malony, A.D., Shende, S., Morris, A.: TAUg: Runtime Global Performance Data Access Using MPI. In: Euro-PVM-MPI (2006).
- [15] : de St. Germain, J.D., Parker, S.G., McCorquodale, J., Johnson, C.R.: Uintah: A massively parallel problem solving environment. In: HPDC'00: International Symposium on High Performance Distributed Computing, pp. 33–42 (2000).

Questions?

- Thank you for attending the talk.
- Email: anataraj (AT) cs.uoregon.edu