Profiling Non-Numeric OpenSHMEM Applications with the TAU Performance System

John C. Linford, Tyler A. Simon, Sameer Shende, Allen D. Malony
{jlinford, tasimon, sameer, malony}@paratools.com
ParaTools, Inc.

Wednesday 5 March 2014

OpenSHMEM'13/14
Annapolis, MD

Slides Available At:
http://www.paratools.com/openshmem14

Summary

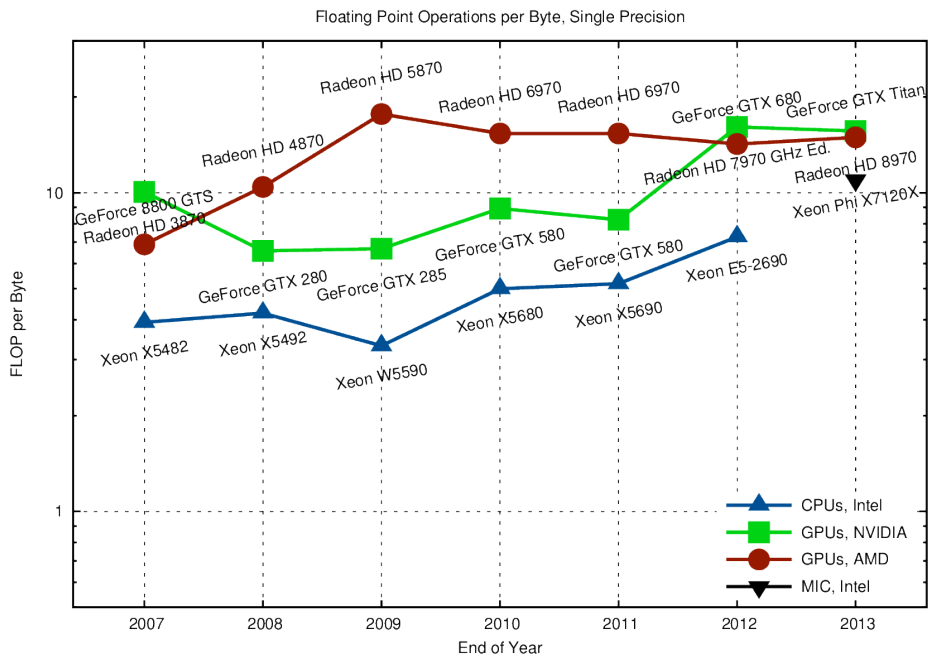
1. Non-Numerical Algorithms
2. Kruskal's Algorithm for Minimum Spanning Tree
3. OpenSHMEM Implementation of Kruskal's Algorithm
4. TAU Performance System support for OpenSHMEM
5. Performance Analysis of OpenSHMEM Kruskal's
6. Conclusion and Acknowledgements

Non-Numerical Algorithms (NNA)

OpenSHMEM and Kruskals's Algorithm for MST

Non-Numerical Algorithms (NNA)

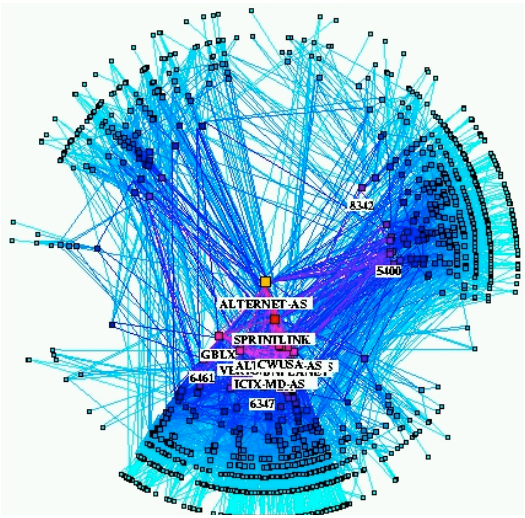
- Low FLOPS/byte ratio.
- Majority of time spent in search or sort.
- Performance depends on **memory locality**.
- Challenging for Top500 HPC systems.



http://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/

Searching or traversing graphs

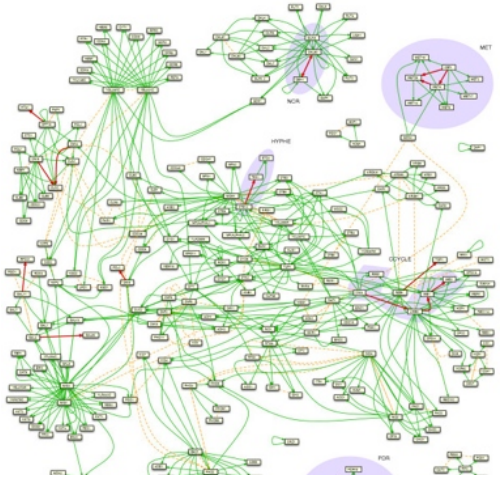
- Connectivity and routing optimization,
- Network design,
- Highway system layout,
- Cluster points in space into natural groups,
- Approximate solutions to hard problems, e.g. TSP



http://nlnet.nl/project/rex/200006-VU-CAIDA/200105-summary/index.html

Motif occurrence in yeast ➔

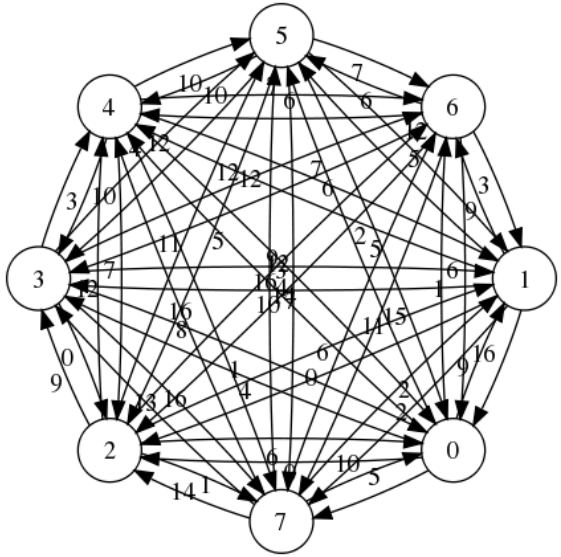
🡐 Network connectivity

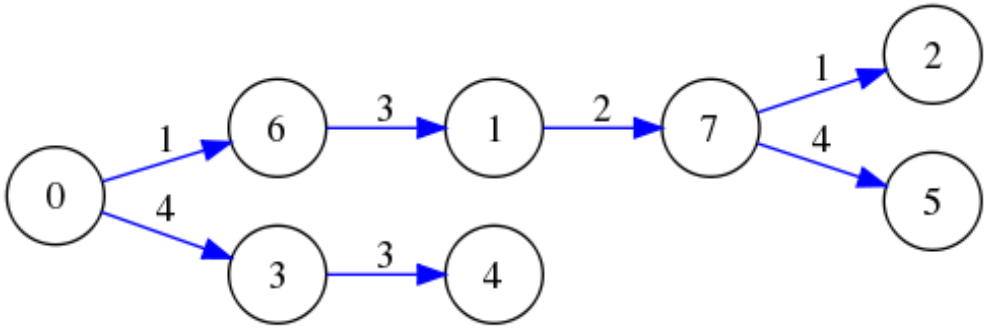


Mazurie et al. Genome Biology 2005 6:R35
doi:10.1186/gb-2005-6-4-r35

Minimum Spanning Trees

- A subgraph that connects all the vertices and is a tree.
- **Kruskal's Algorithm**: choose globally optimal edge (BFS)
- Prim's Algorithm: choose locally optimal edge (DFS)





Minimum Distance = 18

Kruskal's Algorithm

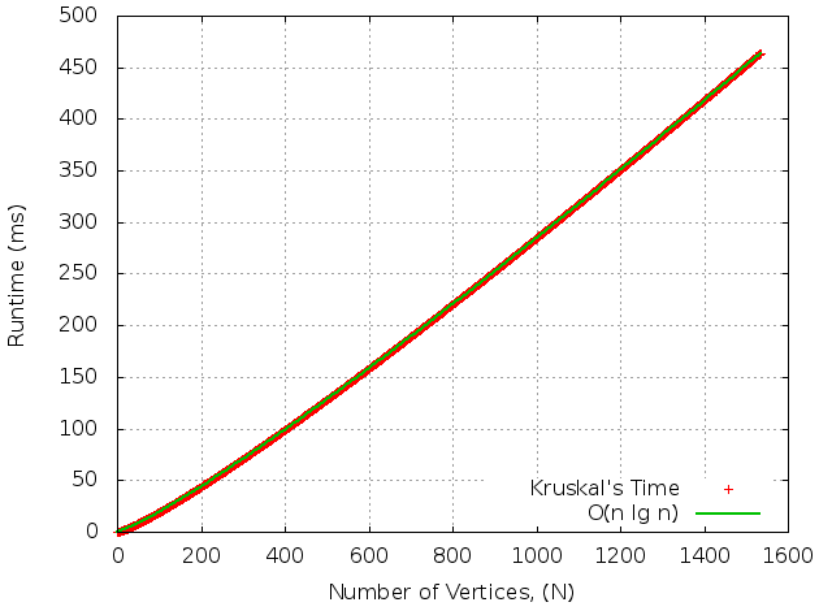
1. Place every vertex in its own set.
2. Select edges in order of increasing weight.
3. As each edge is selected, determine if the vertices connected by that edge are in different sets. If so then insert the edge into the set that is the MST and take the union of sets containing each vertex.
4. Repeat steps 1-3 until all edges have been explored.

In Short:

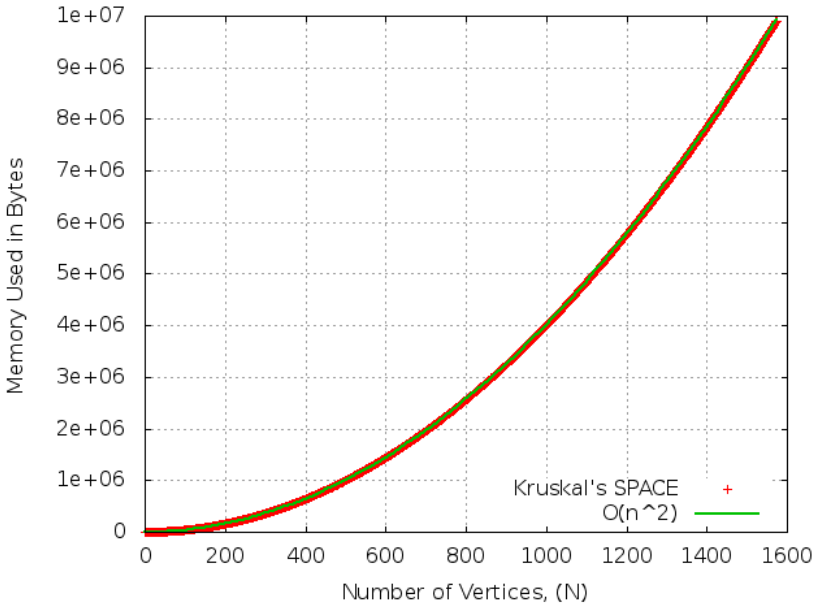
Pick least-weighted edges until the MST contains all nodes.

Complexity of Kruskal's Algorithm

Time Complexity O(n lg n)



Space Complexity O(n^2)



OpenSHMEM Kruskal's Algorithm

```
// Allocate symmetric heap for graph
graph = (int*)shmalloc(graphSize*nNodes*sizeof(int));
span = (int*)shmalloc(graphSize*nNodes*sizeof(int));

// Load and distribute graph from PE_0
... shmem_barrier_all(); ...

// All PEs pick their lowest edges and store to span[]
for (j=0; j<nNodes; ++j) {
  minWeight = INT_MAX;
  minNode = 0;
  for (i=0; i<graphSize; ++i) {
    weight = graph[j*graphSize+i];
    if (weight && (weight < minWeight)) {
      minWeight = weight;
      minNode = i;
    }
  }
  span[j*graphSize+minNode] = minWeight;
}
shmem_barrier_all(); // span[] contains the minimum spanning tree
```

Performance Analysis of Kruskal's Algorithm

TAU Performance System®

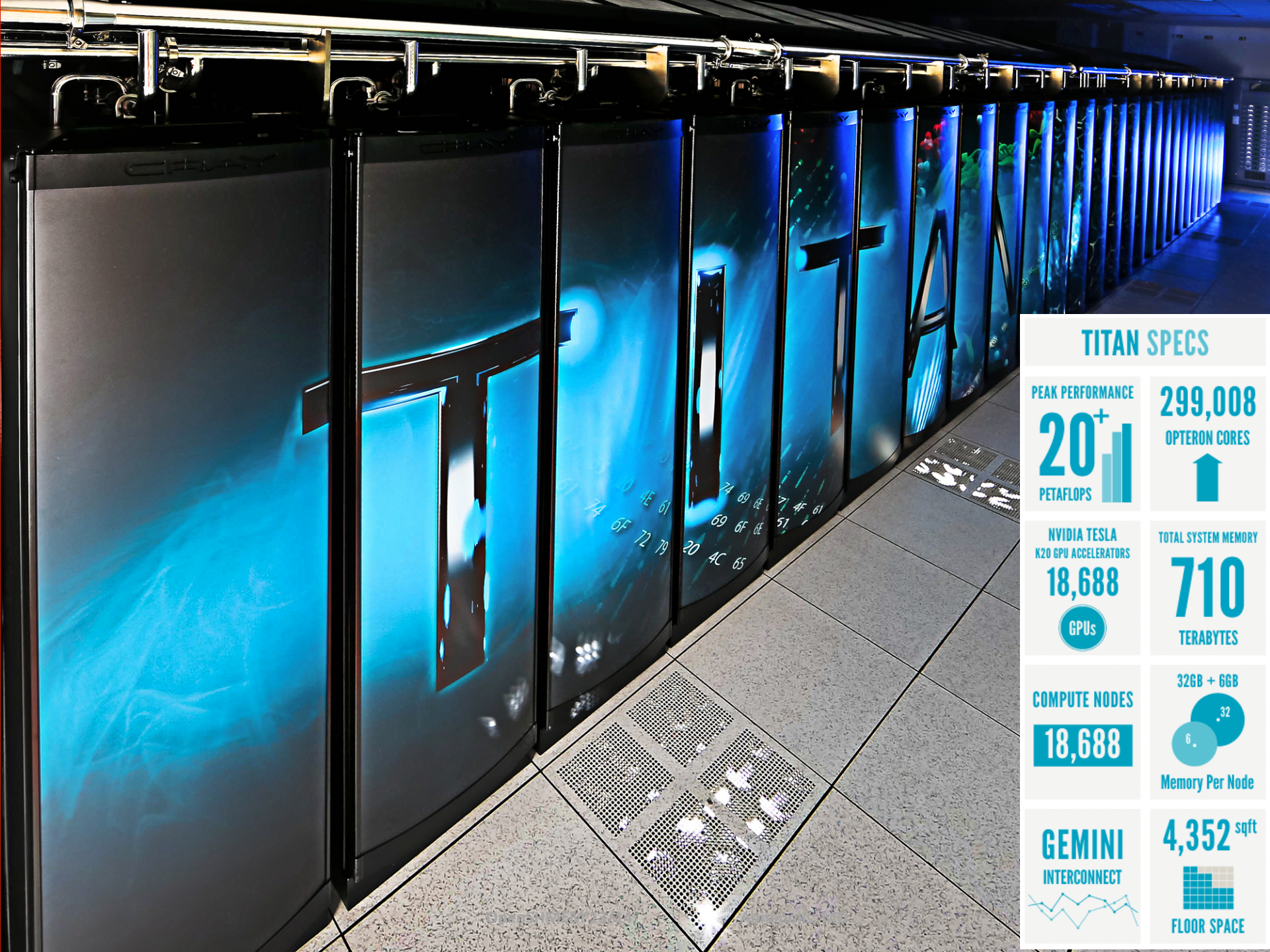
http://tau.uoregon.edu/

- **Tuning and Analysis Utilities (18+ year project)**
- **Comprehensive performance profiling and tracing**
  - Integrated, scalable, flexible, portable
  - Targets all parallel programming/execution paradigms
- **Integrated performance toolkit**
  - Instrumentation, measurement, analysis, visualization
  - Widely-ported performance profiling / tracing system
  - Performance data management and data mining
  - Open source (BSD-style license)
- **Supports OpenSHMEM (and other SHMEM implementations)**

TAU Support for OpenSHMEM

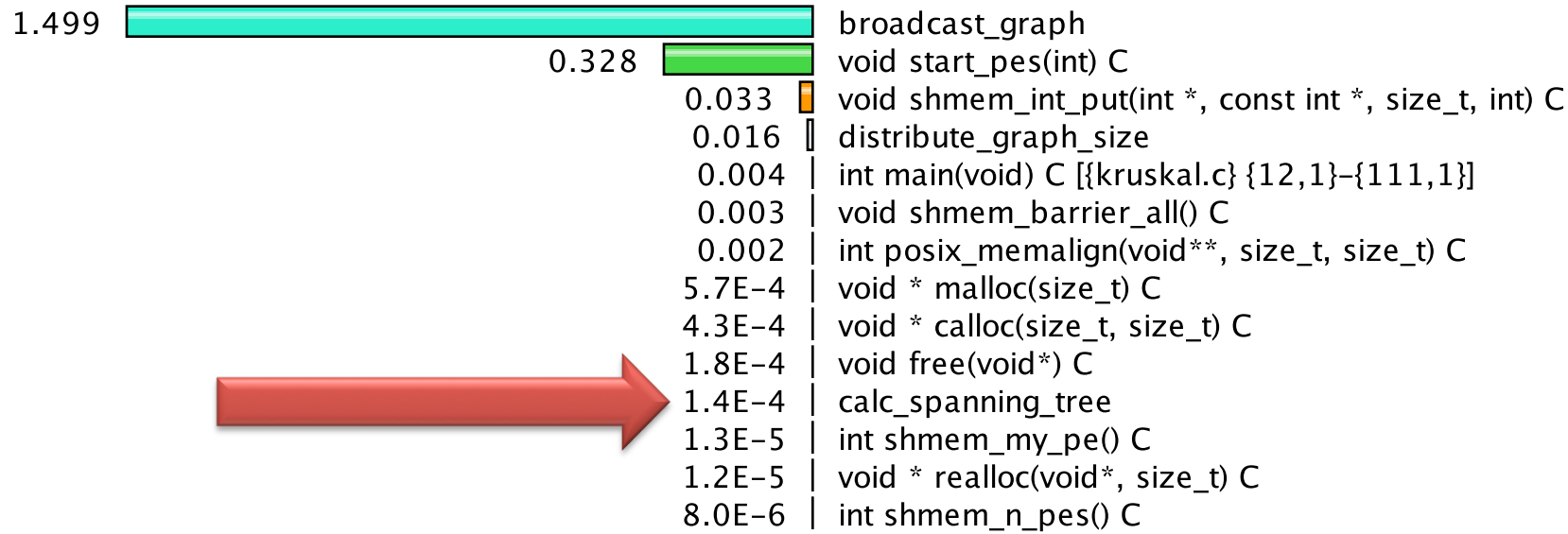
- Source- and compiler-based instrumentation, library interposition, and sampling.
- Track memory allocations in the symmetric heap and in PE local memory.
- Track network I/O caused by transfers to and from symmetric memory.
- Track OpenSHMEM API calls.

Thanks to the OpenSHMEM standard, TAU maintains a single implementation of the SHMEM API wrapper library.


TITAN
TITAN SPECS
PEAK PERFORMANCE
20+
PETAFLOPS
299,008
OPTERON CORES
NVIDIA TESLA
K20 GPU ACCELERATORS
18,688
GPUs
TOTAL SYSTEM MEMORY
710
TERABYTES
COMPUTE NODES
18,688
32GB + 6GB
32
6
Memory Per Node
GEMINI
INTERCONNECT
4,352 sqft
FLOOR SPACE

Performance on ORNL's Titan

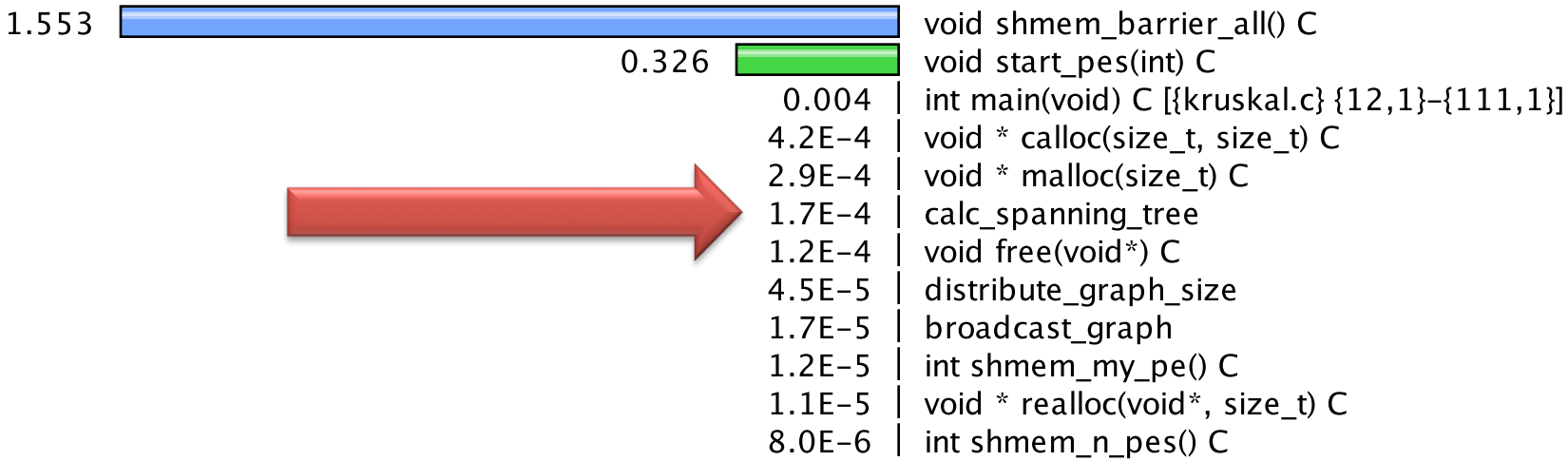
Exclusive seconds on PE 0 of 512 on Titan, showing time spent reading the graph from file and broadcasting it to the other PEs.



1.5 seconds are spent reading the graph from file and broadcasting the edge weights to the other PEs. The time spent calculating the minimum spanning tree is only 140 microseconds.

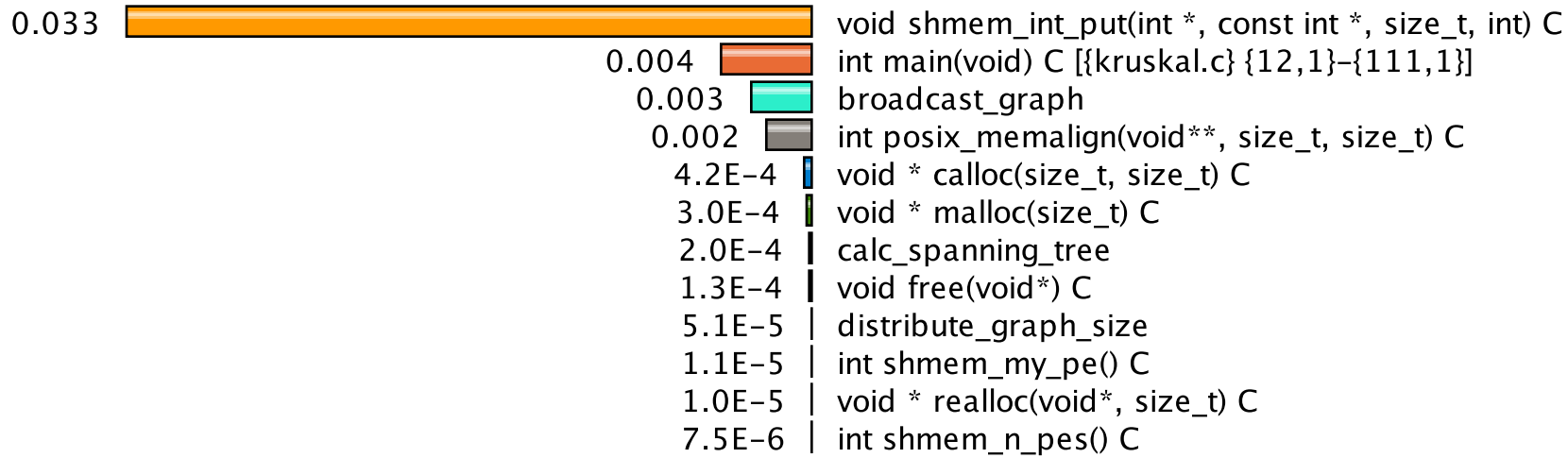
Performance on ORNL's Titan

Exclusive seconds on PE 1 of 512 on Titan, showing time spent waiting to receive the graph data from PE 0 and then calculating the MST.



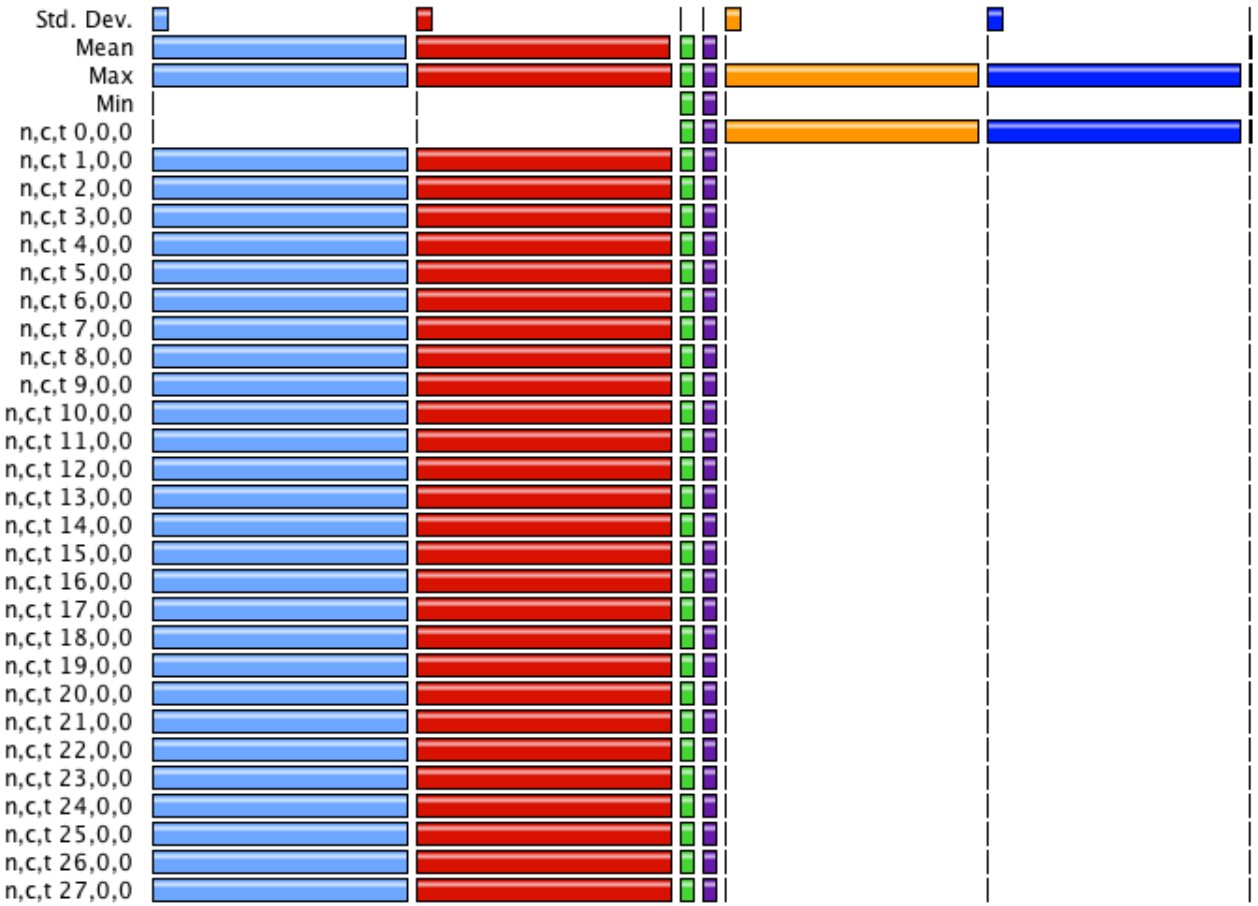
Performance on ORNL's Titan

Exclusive mean time on all 512 PEs on Titan in routines called while calculating the MST after the graph has been broadcast.



Performance on ORNL's Titan

All non-zero PEs have virtually identical performance profiles.



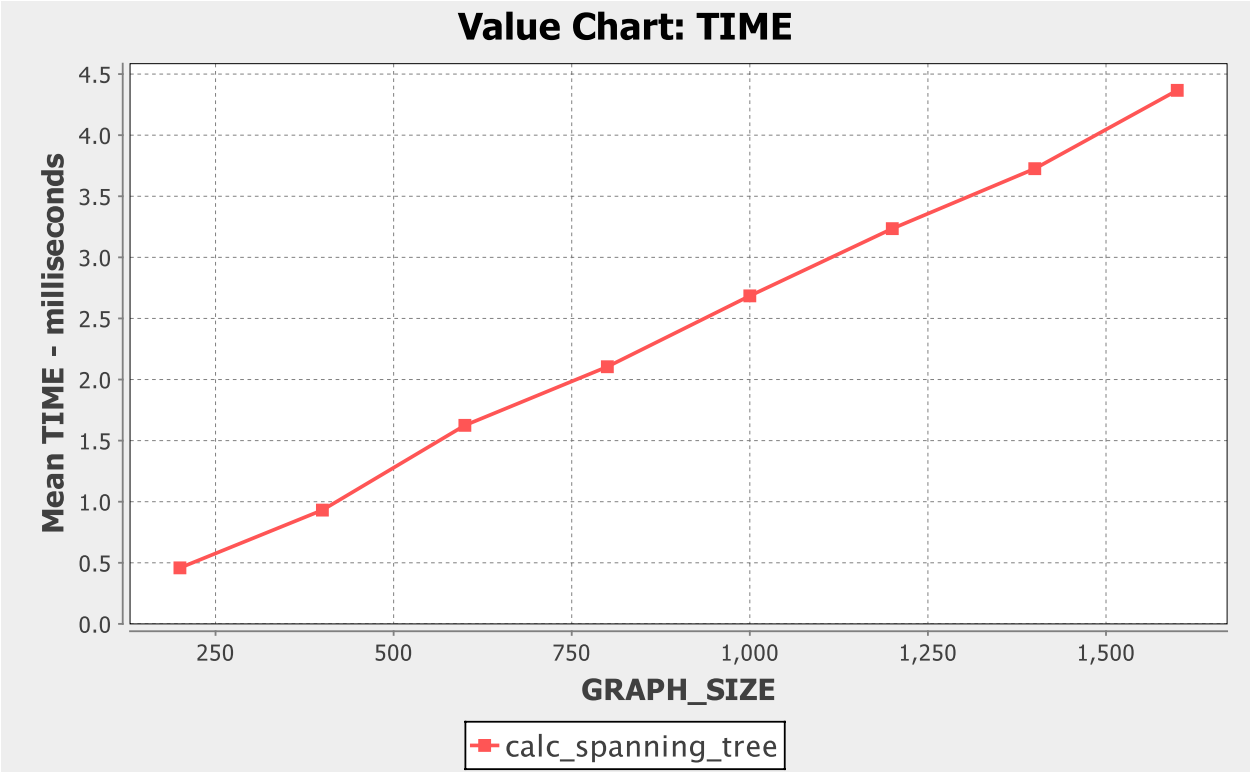
Performance on ORNL's Titan

The mean of context events observed in the OpenSHMEM implementation of Kruskal's algorithm on 512 PEs on Titan.

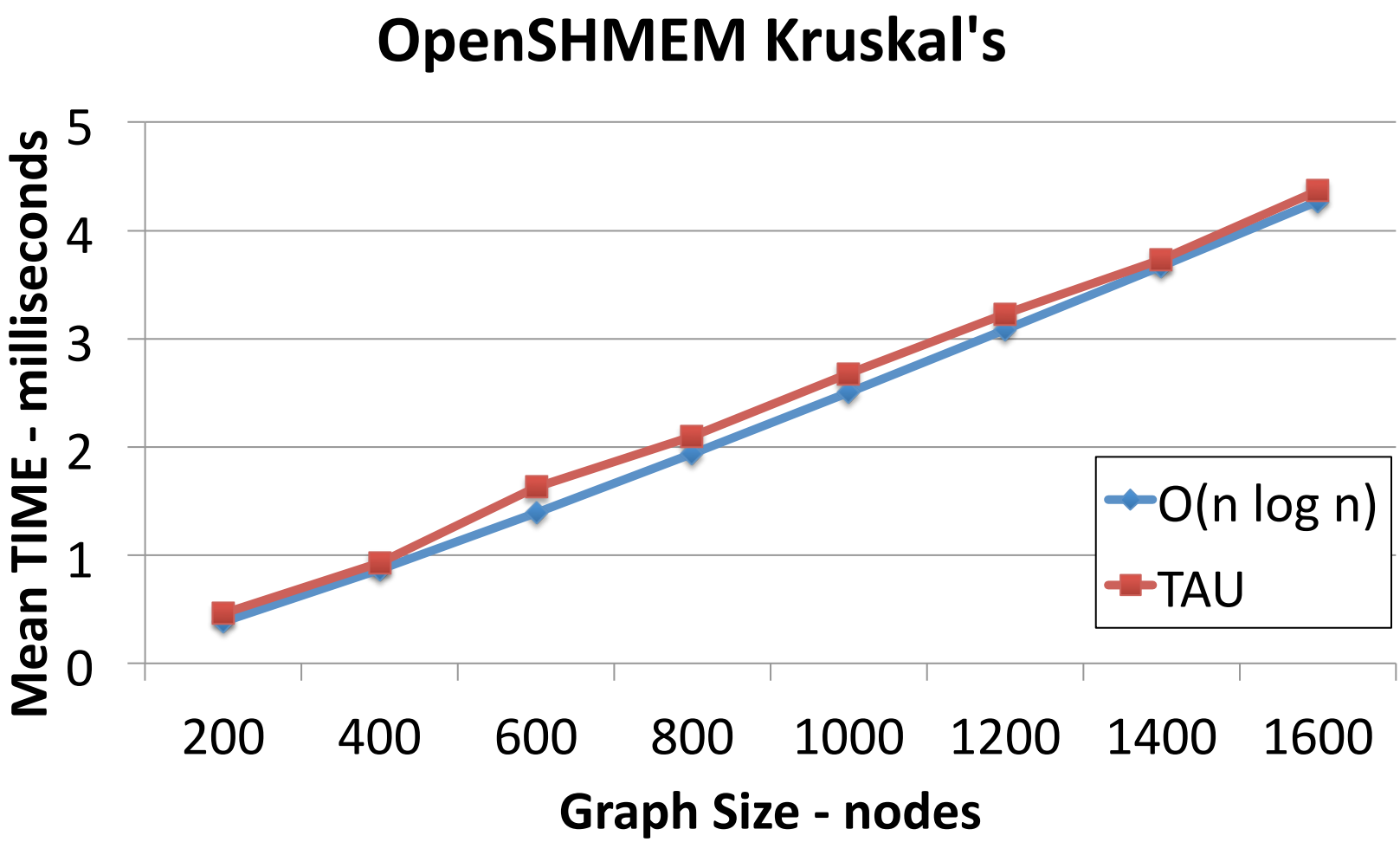
Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
int main(void) C [{kruskal.c} {12,1}-{111,1}]						
void start_pes(int) C						
distribute_graph_size						
Heap Allocate	1,136	2	568	568	568	0
Heap Free	568	1	568	568	568	0
MEMORY LEAK! Heap Allocate	568	1	568	568	568	0
void shmem_int_put(int *, const int *, size_t, int) C						
broadcast_graph						
Heap Allocate	8,192	1	8,192	8,192	8,192	0
Heap Free	8,192	1	8,192	8,192	8,192	0
void shmem_barrier_all() C						
void shmem_int_put(int *, const int *, size_t, int) C						
Heap Allocate	3,204	13	1,728	8	246.462	468.864
Heap Free	1,136	2	568	568	568	0
MEMORY LEAK! Heap Allocate	2,068	11	1,728	8	188	487.43
Message size sent to node 0	8,192	1	8,192	8,192	8,192	0
Message size sent to node 1	8,192	1	8,192	8,192	8,192	0
Message size sent to node 10	8,192	1	8,192	8,192	8,192	0
Message size sent to node 100	8,192	1	8,192	8,192	8,192	0
Message size sent to node 101	8,192	1	8,192	8,192	8,192	0
Message size sent to node 103	8,192	1	8,192	8,192	8,192	0
Message size sent to node 102	8,192	1	8,192	8,192	8,192	0
Message size sent to node 104	8,192	1	8,192	8,192	8,192	0
Message size sent to node 105	8,192	1	8,192	8,192	8,192	0
Message size sent to node 107	8,192	1	8,192	8,192	8,192	0
Message size sent to node 106	8,192	1	8,192	8,192	8,192	0
Message size sent to node 108	8,192	1	8,192	8,192	8,192	0
Message size sent to node 109	8,192	1	8,192	8,192	8,192	0
Message size sent to node 111	8,192	1	8,192	8,192	8,192	0
Message size sent to node 11	8,192	1	8,192	8,192	8,192	0
Message size sent to node 110	8,192	1	8,192	8,192	8,192	0

PerfExplorer: Scalability of Kruskal's

Use **PerfExplorer** (not ParaProf) for multi-profile analysis.

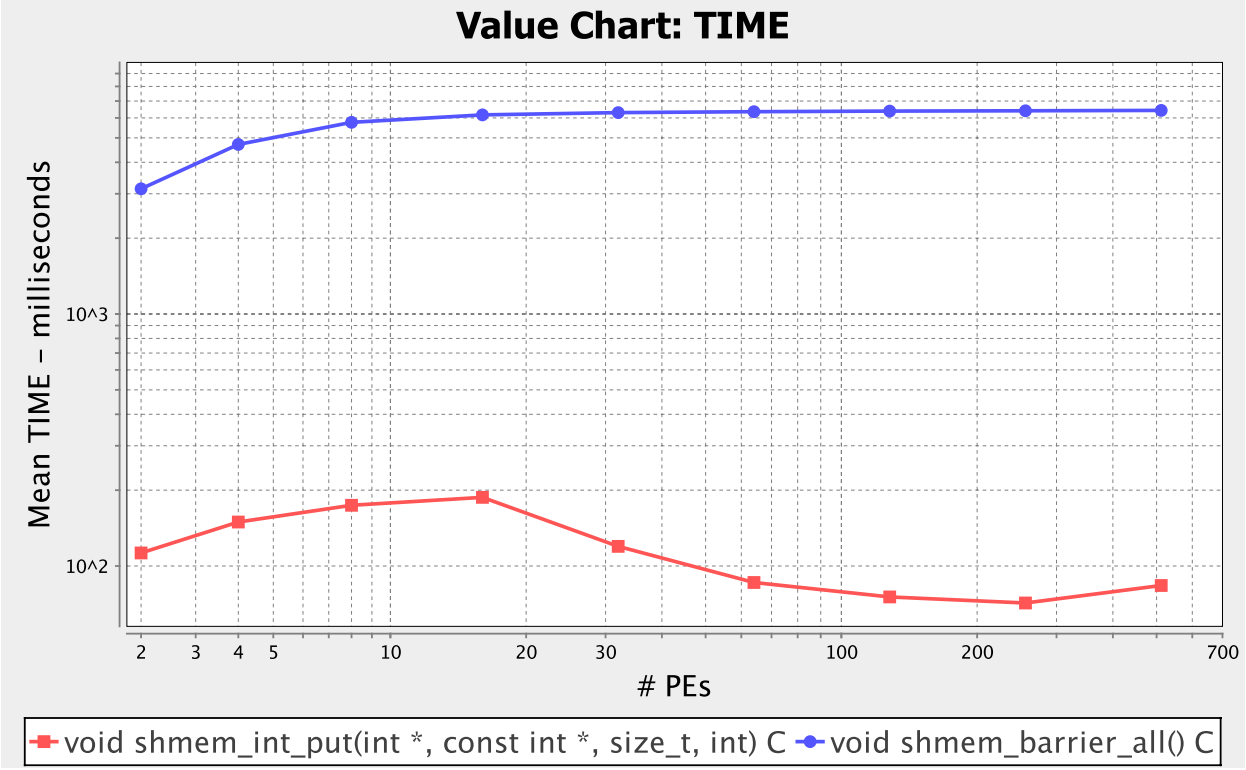


With O(n log n)


OpenSHMEM Kruskal's
Mean TIME - milliseconds
0
1
2
3
4
5
200
400
600
800
1000
1200
1400
1600
Graph Size - nodes
O(n log n)
TAU

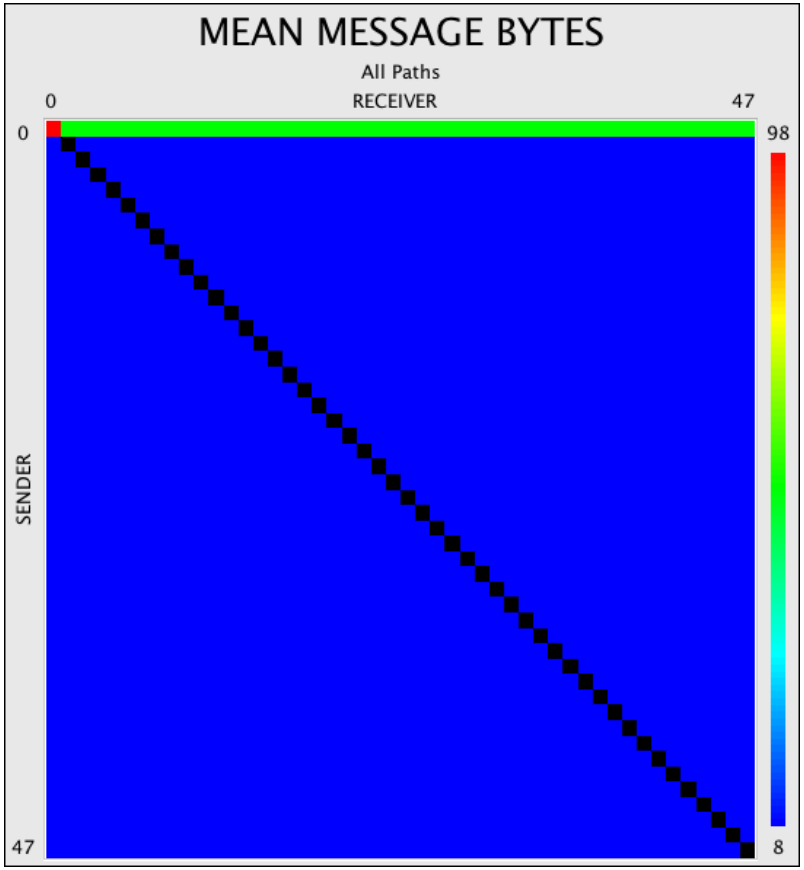
PerfExplorer: Scalability of OpenSHMEM

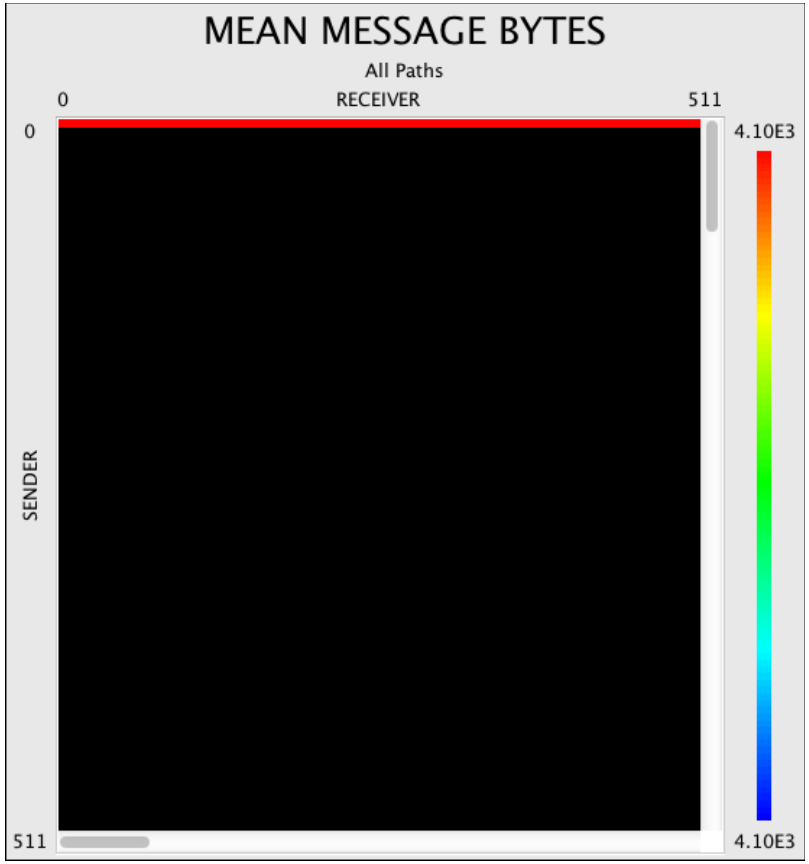
Use **PerfExplorer** (not ParaProf) for multi-profile analysis.



Portability: Performance on UMBC's 48-Core Shared Memory Appliance

Profiles were essentially the same as on Titan except for the comm. matrix





Performance Analysis Overhead

- 4% fully instrumented, a.k.a "pulling out all stops."
- 1.5% if only time in code regions of interest were measured.
  - No memory or I/O measurements.
  - No callpath information in profile.
- TAU's overhead is O(1) in the number of PEs.
- Large applications can benefit from reducing instrumentation to only regions of interest by use of a TAU select file.

Conclusion

- OpenSHMEM's symmetric heap facilitates an effective implementation of Kruskal's Algorithm for MST.
- The TAU Performance System can be used to explore the performance of any OpenSHMEM application with low overhead (1.5-4%)
- OpenSHMEM facilitates portability: our code ran well without modification on two radically different systems.
- Tools like TAU could benefit from further standardization and support for the PSHMEM interface.

Acknowledgements

The University of Oregon NeuroInformatics Center (NIC).

The National Science Foundation Center for Hybrid Multicore Productivity Research at UMBC (CHMPR).

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.