Introspection for Exascale Communication: Needs, Opportunities, Tools and Interfaces

Martin Schulz

Livermore National Laboratory

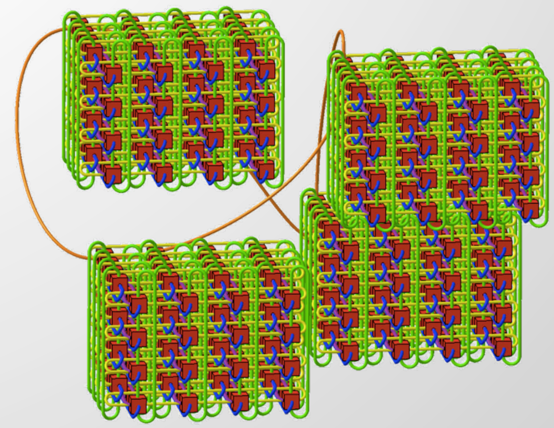
ExaComm Workshop @ ISC 2015 ◆ Frankfurt / Main, July, 2015

LLNL-PRES-674859

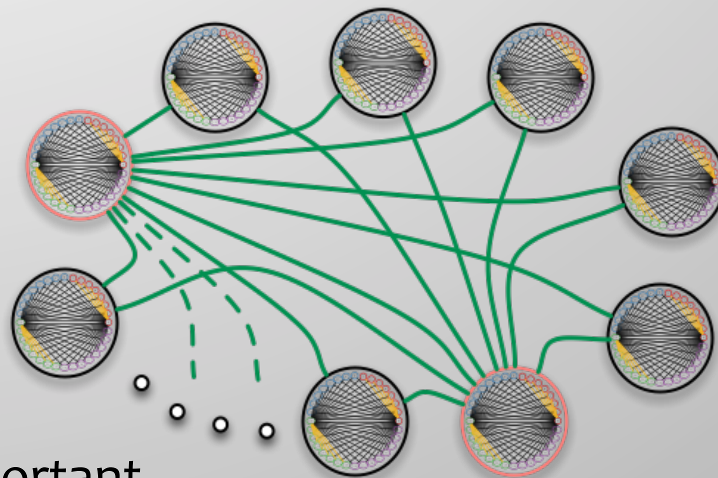
http://scalability.llnl.gov/

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Networks are Getting more Complex



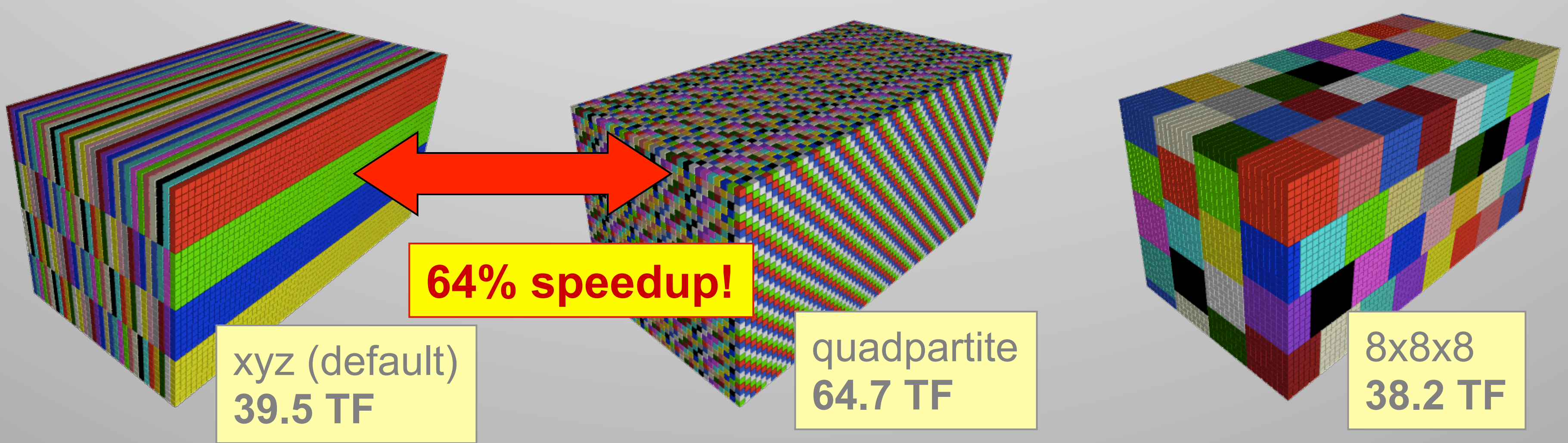
- **More complex network topologies**
  - From 3D torus to 5D torus
  - Complex fat trees, dragonfly networks
  - ➢ Hard to visualize



- **Adaptive Routing**
  - Based on network conditions
  - ➢ Impact of (performance) reproducibility
- **Consequences for Users**
  - Task placements are becoming more important
    - — Even if the vendor tells you otherwise
  - Higher variations in execution time make performance analysis hard
  - Pinpointing contention in the network is almost impossible

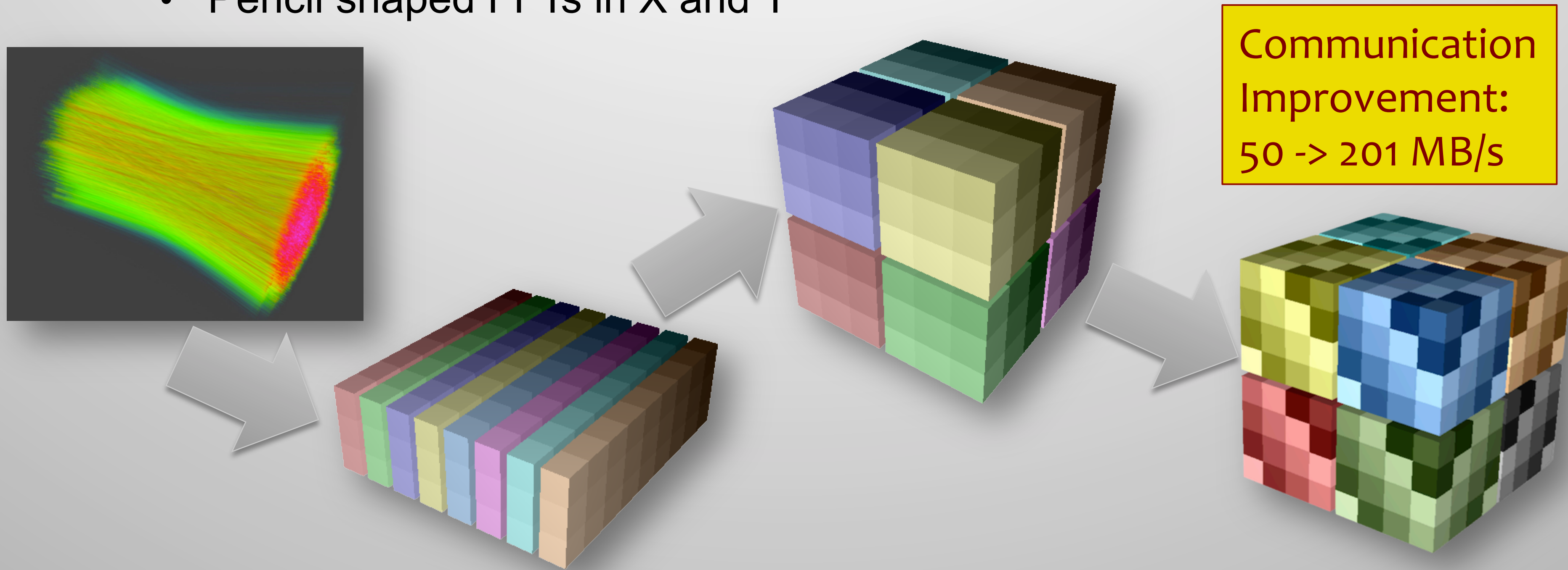
Example: Task Placement

- **Especially for more complex topologies**
  - Interactions with communication topology non-trivial
  - Node placement can have huge impact on performance
- **Example: FPMD Code at LLNL**
  - Target machine: BG/L (3D torus)
  - Full system run at 64K nodes / 128K cores



Plasma/Laser Interaction Code

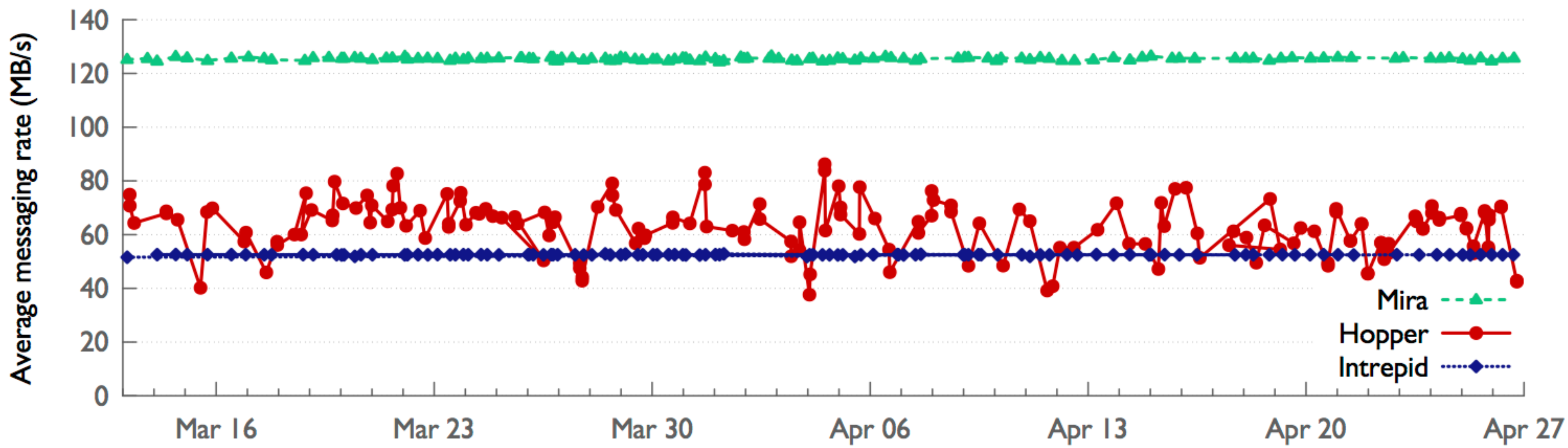
- **3D Communication structure**
  - Stencil communication for Z-coupling
  - Pencil shaped FFTs in X and Y



- **Need for tools to enable such optimizations**
  - Scripting tool to describe task placements in tori: RUBIK
  - Network monitoring to detect link contention

Example: "Cross-run" of Variability

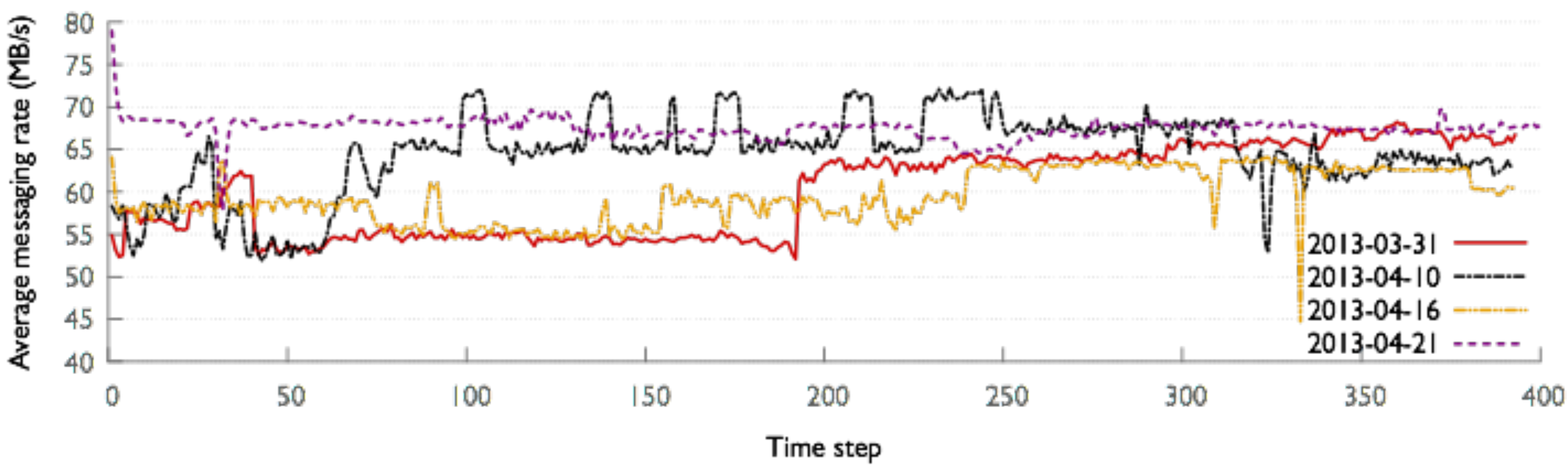
- **Average messaging rates for batch jobs running the same laser-plasma interaction code**
  - Mira: BG/Q (5D torus), Intrepid: BG/P (3D torus)
  - Hopper: Cray XE6 (3D torus)



$$\frac{\text{Total number of bytes sent on the network}}{\text{Time spent sending the messages}}$$

"In Job" Variability

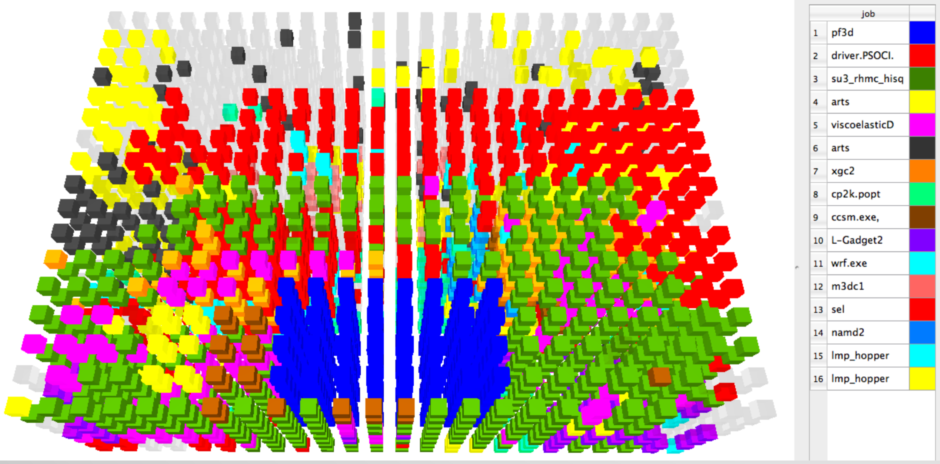
- **Hopper system @ NERSC**
  - Cray XE6 (3D torus)
- **Four long running jobs of the same application**
  - Different dates



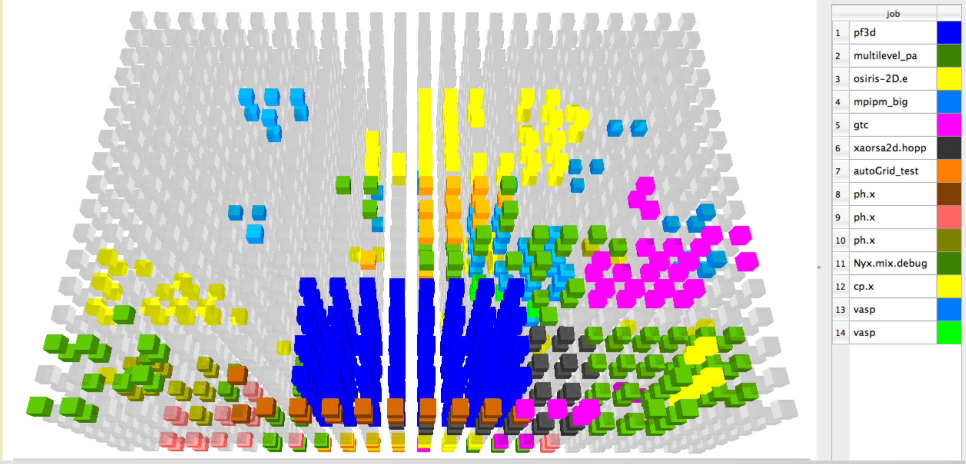
Sources of Variability

- **Operating system noise (OS jitter)**
  - OS daemons running on some cores of each node
- **Placement/location of the allocated nodes for the job (Allocation shape)**
  - Especially problematic if …
    - … not in the hand of the user
    - … partition shapes are not known ahead of time
- **Contention for shared resources (Inter-job contention)**
  - Sharing network links with other jobs

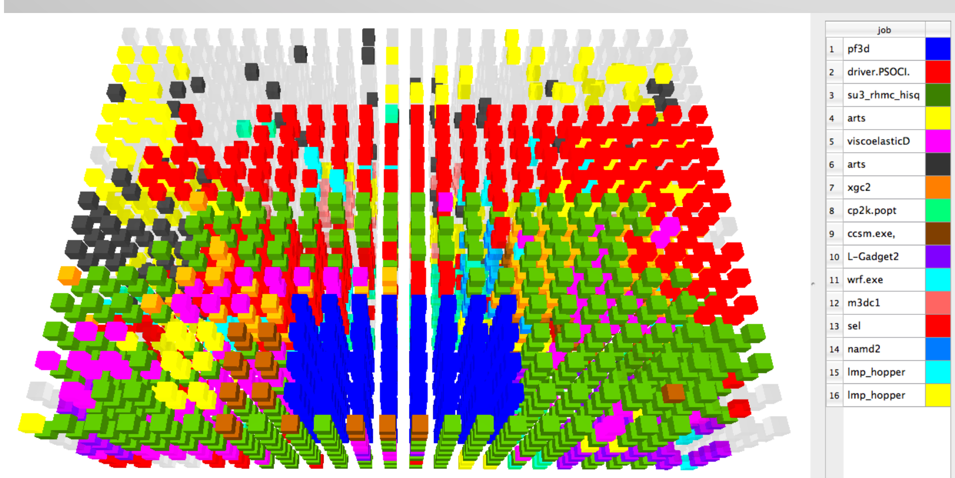
4x8x8-shaped pF3D job (blue)

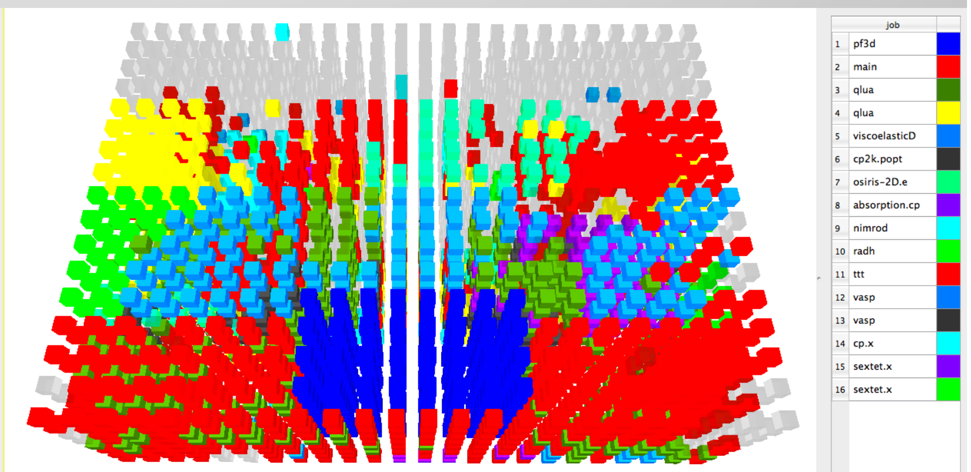


April: 11th: Green = MILC job



April: 16th (a): 25% higher message rate

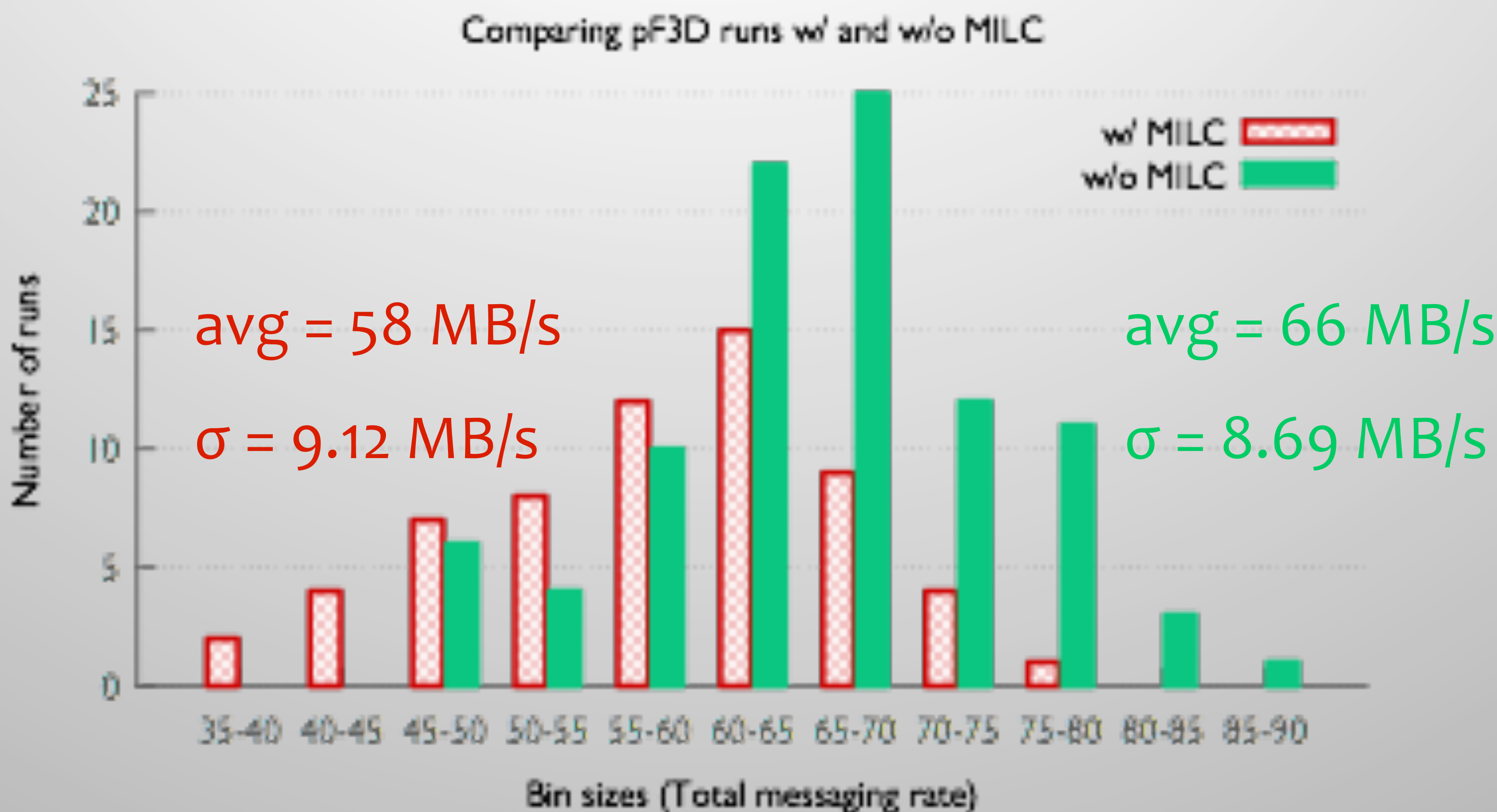




April: 16th (b): red = LSMS job

27.8% higher message rate

Effect of MILC on Laser Code



There goes the neighborhood:
performance degradation due to nearby jobs
Bhatele et al., SC13

Need for Introspection

- **Users need feedback on performance**
  - How are their applications performing?
  - Why are we seeing particular behaviors?
    - — Is it caused by the application or the system?
- **We need efficient / intuitive ways to present this information**
  - Aggregated profiles often don’t tell the whole story
  - ➢ Need for performance visualization
- **Questions**
  - What can we measure today?
  - How can we present it to the user?
  - What would we like to measure?
  - How can we access measured data?

What we Can Measure/Analyze Now?

- **Many networks have performance counters**
  - Packet counts, token counts, some access to switch counters
  - But:
    - Often hard to get to, especially from user level
    - Only accumulative (not per message)
    - Can be expensive (network pertubation)
- **Application level information**
  - MPI interceptors and profilers
  - Useful for correlation to source code
- **Boxfish: A Tool For Network Visualization**
  - Flexible plugins for different topologies
  - Mapping of measurements to topology data

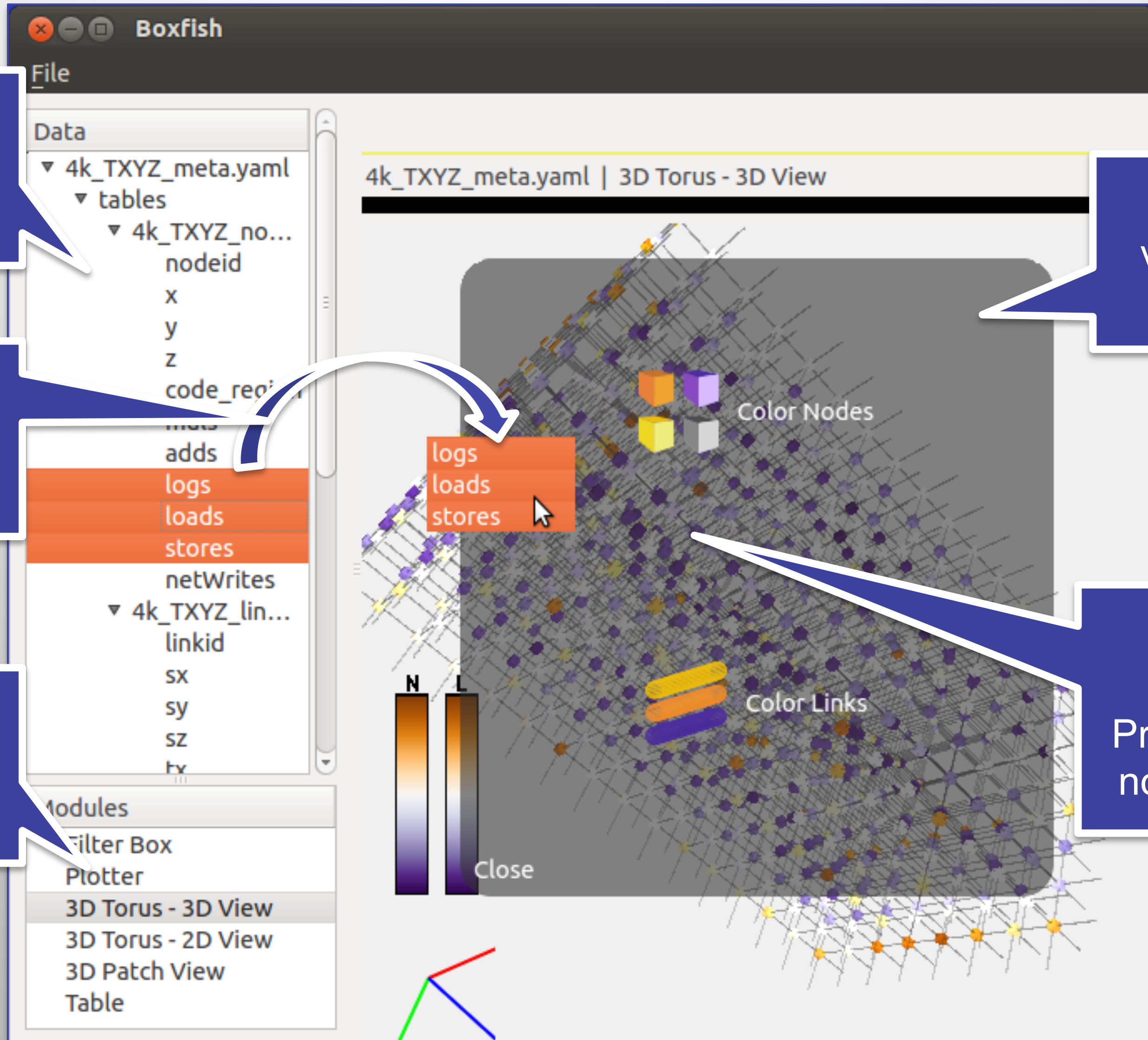
The Boxfish GUI

http://scalability.llnl.gov/

Input data from multiple measurements

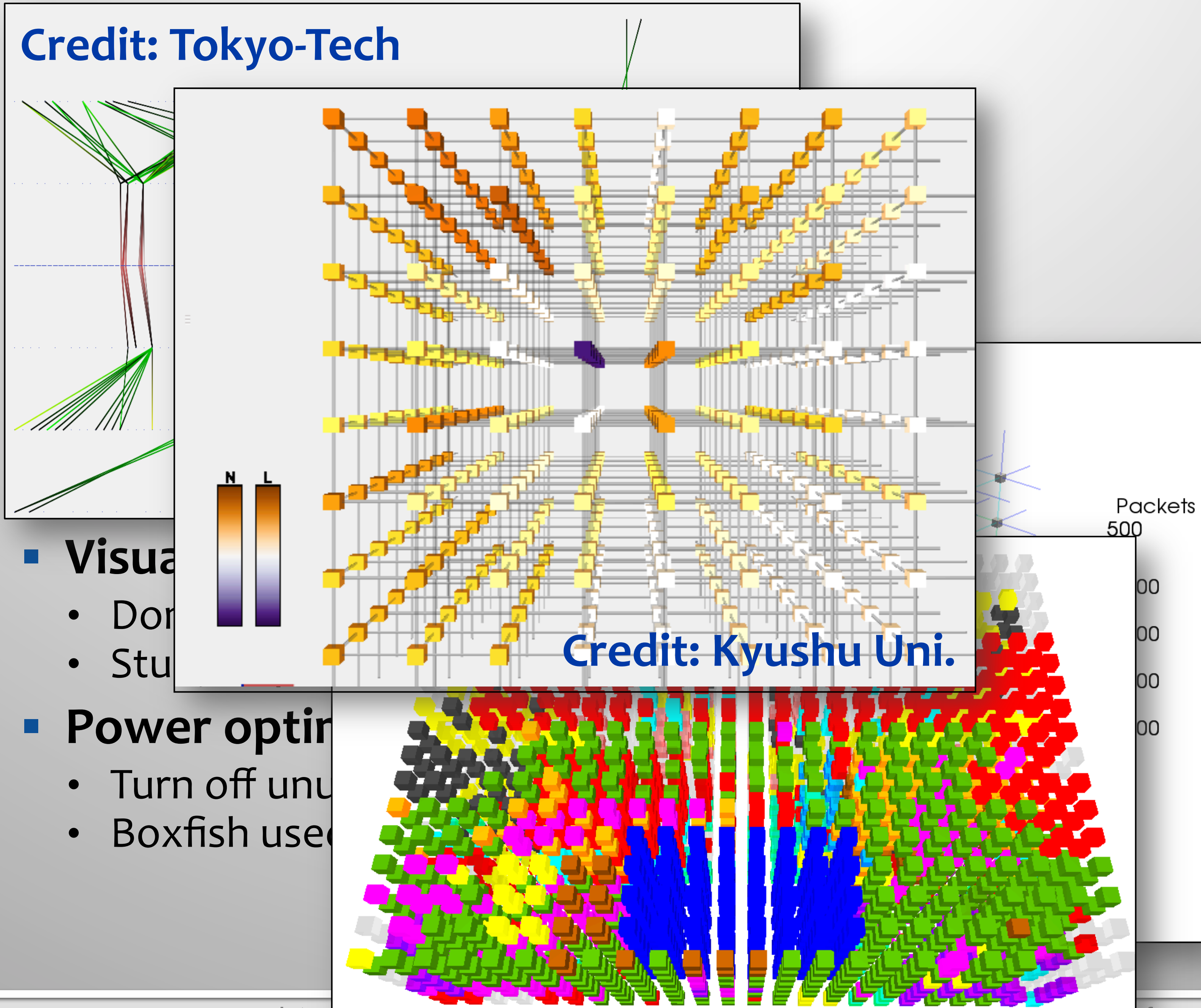
Drag selection to map data to visualization

Available Visualization domains



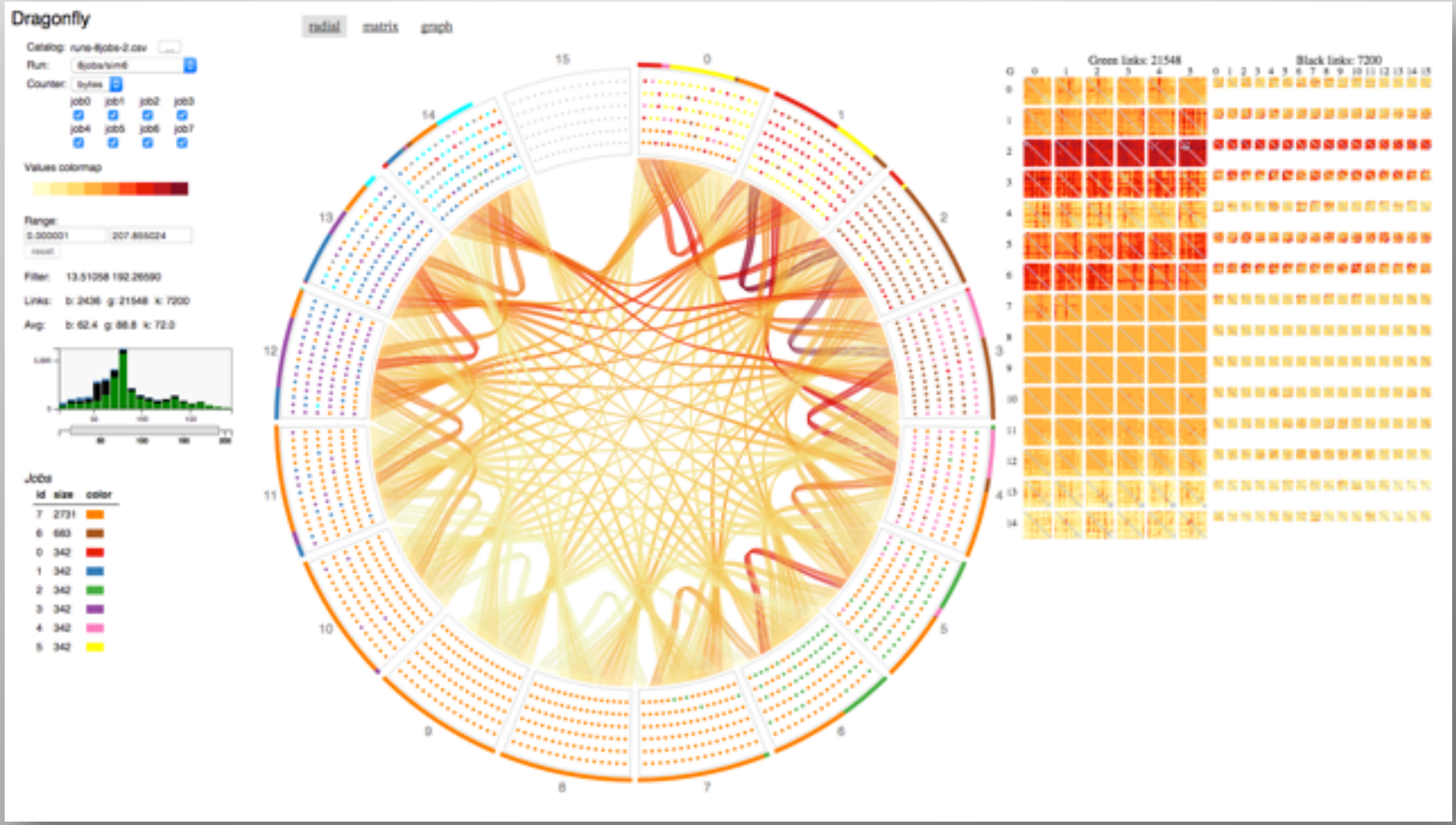
Selected visualization: 3D Torus

Choice of mappings: Present data on nodes or links?

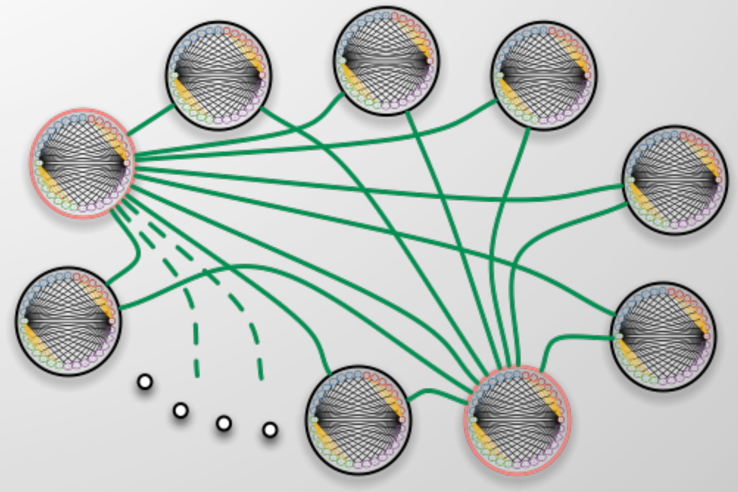


- **Visua**
  - Dor
  - Stu
- **Power optir**
  - Turn off unu
  - Boxfish use

A First Look at a Dragonfly Network



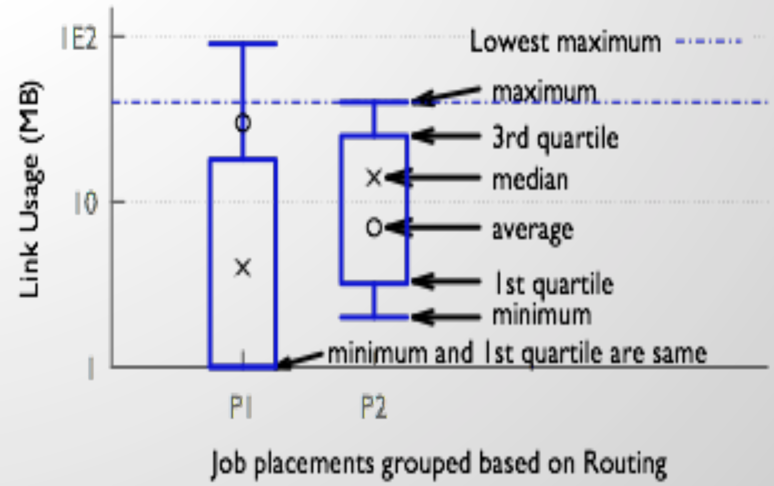
Dragonfly Performance Prediction Model

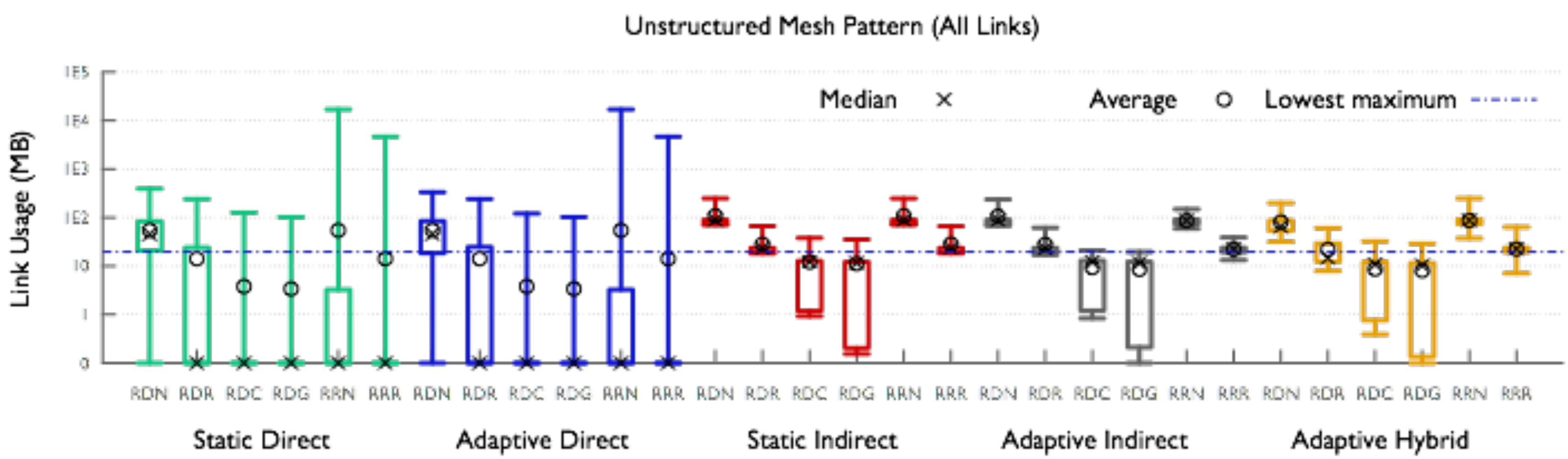


- **Input:**
  - Network graph of dragonfly routers
  - Application communication graph
  - Job placement
  - Routing strategy
- **Output:**
  - The steady-state traffic distribution on all network links, which is representative of the network throughput
- **Routing options**
  - Static Direct (SD): even split among shortest paths
  - Static Indirect (SI): each packet sent on random long path
  - Adaptive Direct (AD): pick least congested shortest path
  - Adaptive Indirect (AI): pick least congested long path
  - Adaptive Hybrid (AH): adaptively pick between AD and AH
- **Different job placement options**

Unstructured Mesh Benchmark

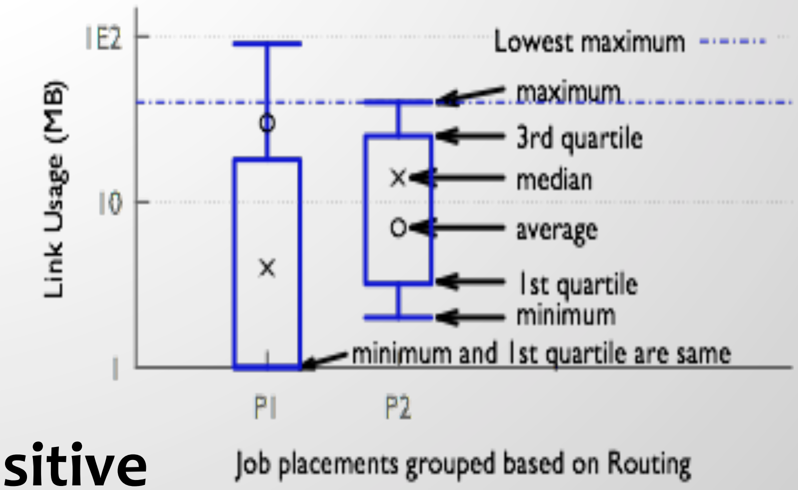
- Adaptive similar to static
- Node placement matters
- Indirect methods slightly worse
  - But less spread
- Hybrid method slightly worse

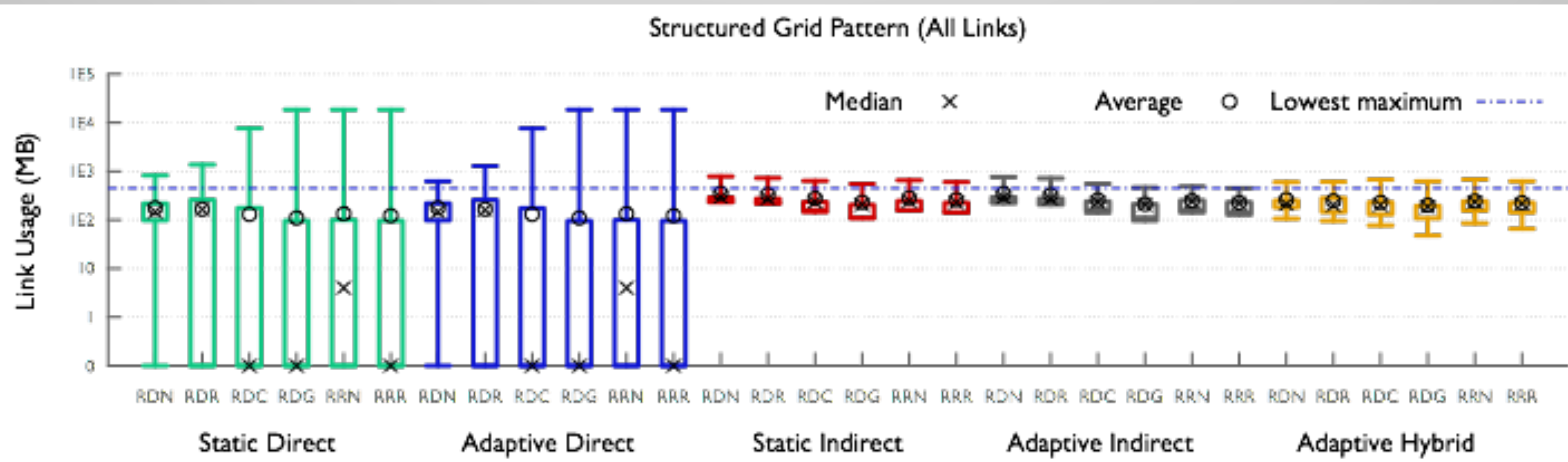




Structured Grid Benchmark

- **Direct methods more variable**
  - Indirect methods take skew away
- **Indirect methods placement insensitive**
- **Adaptive routing slightly better**
- **Hybrid routing slightly worse**





What is Missing? – A Wishlist

- **Dragonfly data so far from simulation**
  - Would need explicit counters for measurements
  - Need access for whole machine counters
    - Links are shared resources across the whole system
    - Influence by system parts on which a job doesn't execute
  - Need data along the message path
- **Need to distinguish traffic by job**
  - Coloring of packets
  - Separate counters
- **Ideally: message tracking**
  - Color individual messages
  - Count impact of other messages on a colored messages
  - Would allow …
    - … precise location of contention spots
    - … clean correlation with actual application messages

How to Get to The Information?

- **State of the Art**
  - Many vendor specific solutions
    - In rare cases through PAPI, but even then hardware specific
    - Prohibits portable tools
  - Only light at the end of the tunnel: PMPI
    - MPI Profiling Interface for application level interception
    - Part of MPI since MPI 1.0
- **Need standard interfaces to enable portable tools**
  - Should be part of an existing (pseudo) standard
    - In a language standard would be best
    - PAPI would also be helpful

The MPI Tool Information Interface

- **Short name: the MPI_T interface**
  - Provide introspection into the MPI layer
  - Export internal performance information
  - Enable standardized access for tools
  - Allow for implementation specific information
- **Included in MPI 3.0 as part of a new tools chapter**
  - Replaces the existing MPI profiling interface chapter
  - PMPI included as a new subchapter (unchanged)
- **API is a set of routines with the prefix MPI_T**
  - Mostly encapsulated in the new chapter
  - Same general restrictions/requirements as any MPI routine
  - Ability to use MPI_T before MPI_Init / after MPI_Finalize

General Approach

- **Basic concept: a set of named variables**
  - Set of variables and naming left to the MPI implementation
  - MPI_T provides query functions to detect variables
  - Semantics provided as clear text
  - Routines to read and write values of these variable
- **Split into performance and control variables**
  - Performance: internal performance data
    - — "Software counters for MPI"
  - Control: Configuration information / environment variables

Performance Variables
- Number of packets sent
- Time spent blocking
- Memory allocated

Control Variables
- Parameters like Eager Limit
- Startup control
- Buffer sizes and management

Basic Functionality for Variables

Performance Variables

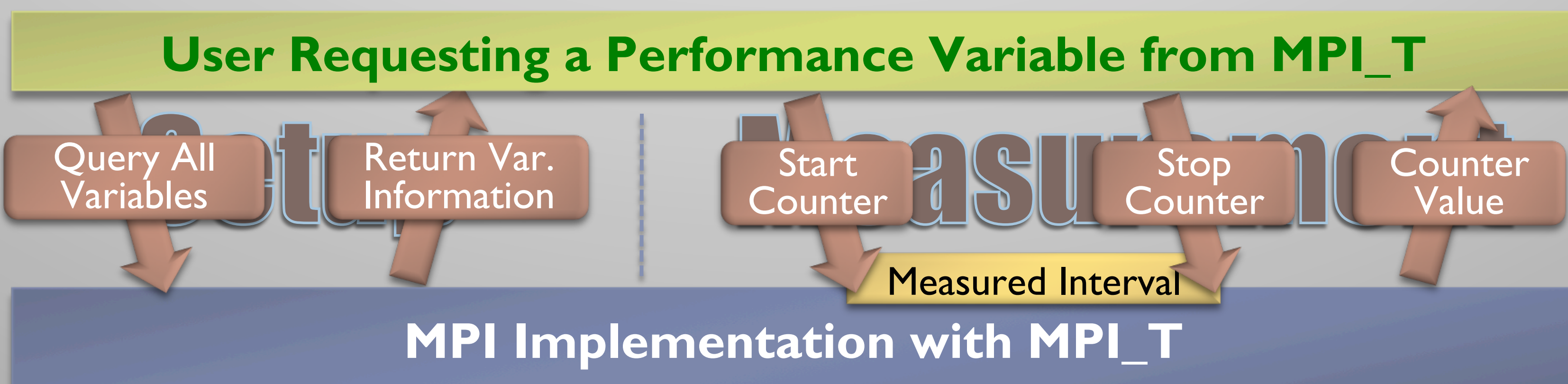
- Allocate Session
- Allocate Handle
- **Reset/Write Variable**
- **Start Variable**
- **Stop Variable**
- **Read/Readreset Variable**
- Free Handle
- Free Session

Control Variables

- Allocate Handle
- **Read/Write Variable**
  - Scoping to define to which ranks a configuration change must be applied to
- Free Handle

MPI_T Query Approach

- **Which variables exist can differ between …**
  - … MPI implementations
  - … compilations of the MPI library (debug vs. production version)
  - … executions of the same application/MPI library
  - Libraries can decide not to provide any variables
- **Example: Performance Variables:**
  - Users need to query existing variables
  - MPI provides names, descriptions, and metadata



Making Use of the MPI_T Interface

- **Typical Chicken & Egg problem**
  - Tools need the interface to be available
  - Library developers need to see the need
  - Test cases during development
- **Two initial tools for MPI_T**
  - — Varlist: list available options (lists existing variables)
  - — Gyan: profiling using MPI_T Variables (basic end-to-end profiler)

  http://scalability.llnl.gov/

- **Other early approaches**
  - Use of MPI_T control variables for auto-tuning (part of Periscope)
  - Discussions for a PAPI integration

Varlist: Querying Existing Variables

- **Simple tool to read all variables offered**
  - Extract descriptions and metadata
  - Read default values

OPTIONS	DESCRIPTION
-c	List only Control Variables
-p	List only Performance Variables
-v <VL>	List up to verbosity level=[1,9]
-l	Long list with all information, including descriptions
-m	Do not call `MPI_Init` before listing variables

- **Use cases**
  - Gather information about which variables are available
  - Documentation of runtime environment

Example: Control Variables on Open MPI

```
================================
Control Variables
================================

Found 1026 control variables
Found 1026 control variables with verbosity <= D/A-9

Variable                                              VRB   Type  Bind    Scope    Value
-----------------------------------------------------------------------------------------------------
...
mpi_ddt_unpack_debug                                  U/A-3 INT   n/a     LOCAL    false
mpi_ddt_pack_debug                                    U/A-3 INT   n/a     LOCAL    false
mpi_ddt_position_debug                                U/A-3 INT   n/a     LOCAL    false
mpi_ddt_copy_debug                                    U/A-3 INT   n/a     LOCAL    false
dss_buffer_type                                       D/D-8 INT   n/a     ALL      described
dss_buffer_initial_size                               D/D-8 INT   n/a     ALL      128
dss_buffer_threshold_size                             D/D-8 INT   n/a     ALL      1024
event                                                 U/D-2 CHAR  n/a     ALL
event_base_verbose                                    D/D-8 INT   n/a     LOCAL    0
event_libevent2021_event_include                      U/A-3 CHAR  n/a     LOCAL    poll
opal_event_include                                    U/A-3 CHAR  n/a     LOCAL    poll
event_libevent2021_major_version                      D/A-9 INT   n/a     UNKNOWN  1
event_libevent2021_minor_version                      D/A-9 INT   n/a     UNKNOWN  9
event_libevent2021_release_version                    D/A-9 INT   n/a     UNKNOWN  0
mpi_param_check                                       D/A-9 INT   n/a     READONLY true
mpi_yield_when_idle                                   D/A-9 INT   n/a     READONLY false
mpi_event_tick_rate                                   D/A-9 INT   n/a     READONLY -1
mpi_show_handle_leaks                                 D/A-9 INT   n/a     READONLY true
mpi_no_free_handles                                   D/A-9 INT   n/a     READONLY false
mpi_show_mpi_alloc_mem_leaks                          D/A-9 INT   n/a     READONLY 0
mpi_show_mca_params                                   D/A-9 CHAR  n/a     READONLY
mpi_show_mca_params_file                              D/A-9 CHAR  n/a     READONLY
mpi_abort_delay                                       D/A-9 INT   n/a     READONLY 0
mpi_abort_print_stack                                 D/A-9 INT   n/a     READONLY true
...
```

Example: Performance Variables in MVAPICH

```
===============================
Performance Variables
===============================

Found 25 performance variables
Found 25 performance variables with verbosity <= D/A-9

Variable                                   VRB   Class   Type   Bind     R/O CNT ATM
------------------------------------------------------------------------------------
posted_recvq_length                        U/D-2 LEVEL   UINT   n/a      YES YES  NO
unexpected_recvq_length                    U/D-2 LEVEL   UINT   n/a      YES YES  NO
posted_recvq_match_attempts                U/D-2 COUNTER UNKNOW n/a       NO YES  NO
unexpected_recvq_match_attempts            U/D-2 COUNTER UNKNOW n/a       NO YES  NO
time_failed_matching_postedq               U/D-2 TIMER   DOUBLE n/a       NO YES  NO
time_matching_unexpectedq                  U/D-2 TIMER   DOUBLE n/a       NO YES  NO
unexpected_recvq_buffer_size               U/D-2 LEVEL   UNKNOW n/a      YES YES  NO
mem_allocated                              U/B-1 LEVEL   ULLONG n/a      YES YES  NO
mem_allocated                              U/B-1 HIGHWAT ULLONG n/a      YES YES  NO
mv2_progress_poll_count                    D/B-7 COUNTER ULONG  n/a       NO  NO  NO
coll_bcast_binomial                        U/B-1 COUNTER ULLONG n/a      YES YES  NO
coll_bcast_scatter_doubling_allgather      U/B-1 COUNTER ULLONG n/a      YES YES  NO
coll_bcast_scatter_ring_allgather          U/B-1 COUNTER ULLONG n/a      YES YES  NO
mv2_num_2level_comm_requests               U/D-2 COUNTER ULONG  n/a      YES YES  NO
mv2_num_2level_comm_success                U/D-2 COUNTER ULONG  n/a      YES YES  NO
mv2_num_shmem_coll_calls                   T/B-4 COUNTER ULONG  n/a      YES YES  NO
mv2_coll_bcast_binomial                    T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_doubling_allgather  T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_ring_allgather      T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_scatter_ring_allgather_shm  T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_shmem                       T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_knomial_internode           T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_knomial_intranode           T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_mcast_internode             T/B-4 COUNTER ULLONG n/a      YES YES  NO
mv2_coll_bcast_pipelined                   T/B-4 COUNTER ULLONG n/a      YES YES  NO
------------------------------------------------------------------------------------
```

Gyan = Knowledge

- **Basic tool to profile MPI_T information**
  - Calipers for whole program execution
  - Predefined counters defined through environment variable
    - Identify with Varlist
  - Alternatively: monitor all available variables
- **Implemented as a PMPI tool**
  - Transparent preloading
  - Data collected and printed at the end of execution
- **Following experiments**
  - LLNL TLCC cluster (Dual socket Intel Sandybridge nodes and IB)
  - MVAPICH2-2.0a

Gyan Output

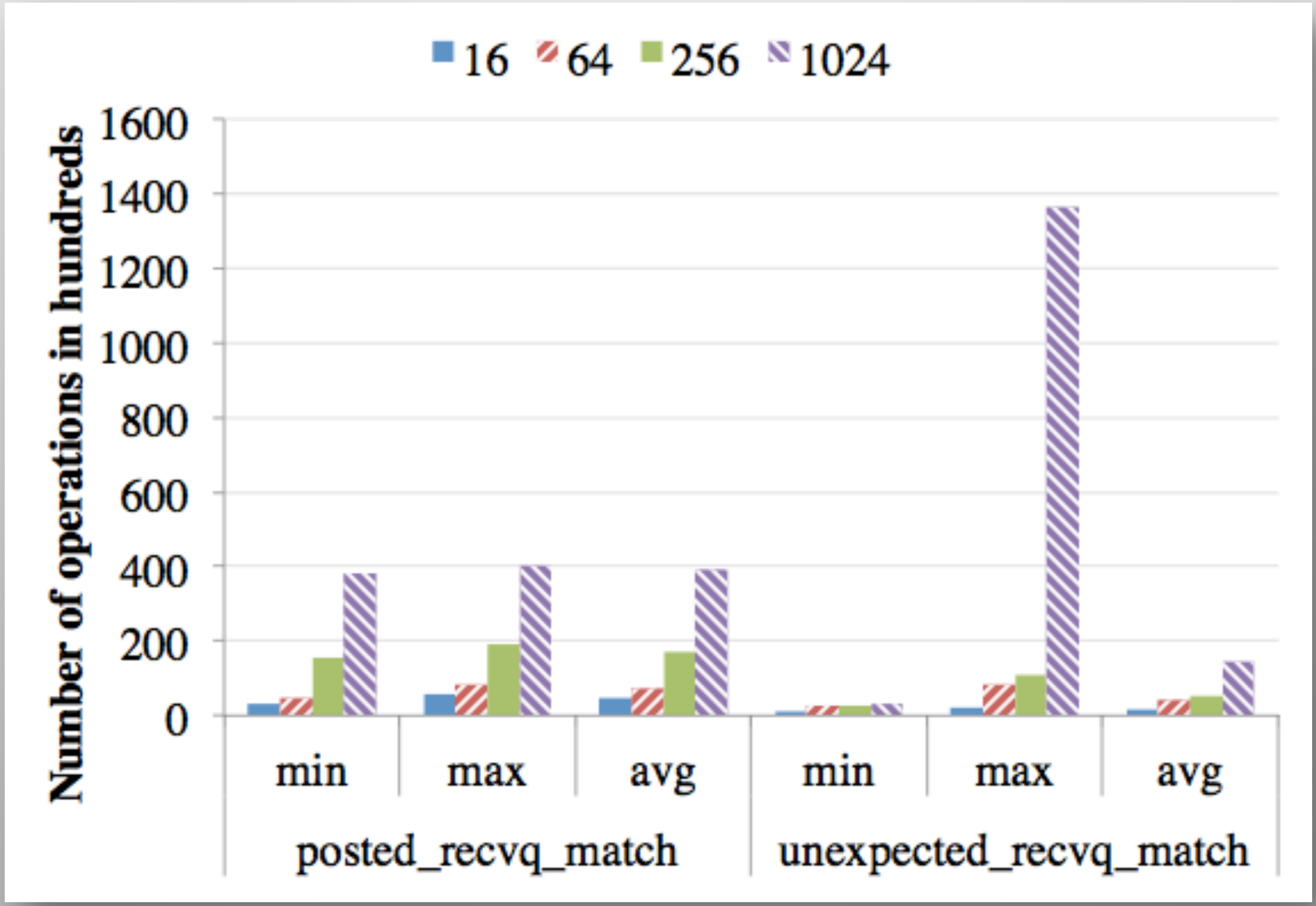
Performance profiling for the complete MPI job:

Variable Name	Type	Minimum	Maximum	Average
mem_allocated	LEVEL	2119601	2119601	2119601.00
mem_allocated	HIGHWAT	17488028	17488028	17488028.00
mv2_reg_cache_hits	COUNTER	1205	1205	1205.00
mv2_reg_cache_misses	COUNTER	5	5	5.00
mv2_vbuf_allocated	COUNTER	384	384	384.00
mv2_vbuf_freed	COUNTER	160085	160085	160085.00
mv2_vbuf_available	COUNTER	283	283	283.00
mv2_ud_vbuf_n_allocated	COUNTER	0	0	0.00
mv2_ud_vuf_freed	COUNTER	0	0	0.00
mv2_ud_vbuf_available	COUNTER	0	0	0.00
mv2_progress_poll_count	COUNTER	753207	753207	753207.00
mv2_rdma_ud_retransmit_count	COUNTER	0	0	0.00
coll_bcast_binomial	COUNTER	462	462	462.00
coll_bcast_scatter_doubling_allgather	COUNTER	0	0	0.00
coll_bcast_scatter_ring_allgather	COUNTER	0	0	0.00
mv2_num_2level_comm_requests	COUNTER	1	1	1.00
mv2_num_2level_comm_success	COUNTER	1	1	1.00
mv2_num_shmem_coll_calls	COUNTER	21276	21276	21276.00
mv2_coll_bcast_binomial	COUNTER	0	0	0.00
mv2_coll_bcast_scatter_doubling_allgather	COUNTER	0	0	0.00
mv2_coll_bcast_scatter_ring_allgather	COUNTER	220	220	220.00
mv2_coll_bcast_scatter_ring_allgather_shm	COUNTER	110	110	110.00
mv2_coll_bcast_shmem	COUNTER	3520	3520	3520.00
mv2_coll_bcast_knomial_internode	COUNTER	1760	1760	1760.00
mv2_coll_bcast_knomial_intranode	COUNTER	0	0	0.00
mv2_coll_bcast_mcast_internode	COUNTER	0	0	0.00
mv2_coll_bcast_pipelined	COUNTER	110	110	110.00
mv2_ibv_channel_ctrl_packet_count	COUNTER	0	0	0.00
mv2_ibv_channel_out_of_order_packet_count	COUNTER	0	0	0.00
mv2_ibv_channel_out_of_order_packet_count	COUNTER	0	0	0.00
mv2_rdmafp_ctrl_packet_count	COUNTER	0	0	0.00
mv2_rdmafp_out_of_order_packet_count	COUNTER	0	0	0.00
mv2_rdmafp_exact_recv_count	COUNTER	73245	73245	73245.00
mv2_rdmafp_sendconn_accepted	COUNTER	4	4	4.00

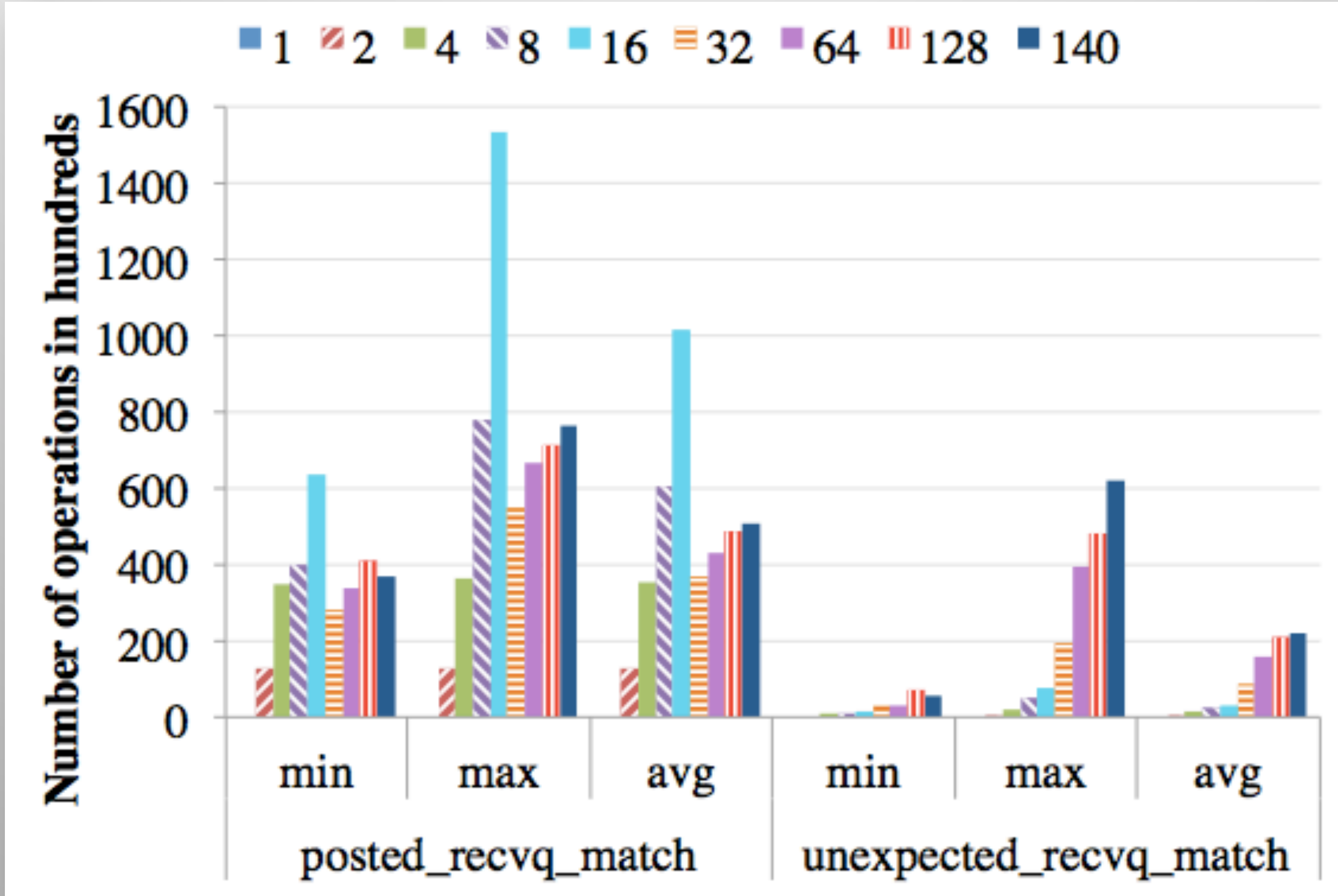
Counters of Interest for Case Studies

Variable	Description
posted_recvq_match	Counts how many times the queue for receiving expected messages is read.
unexpected_recvq_match	Counts how many times the queue for receiving unexpected messages is read.
progress_poll_count	Counts how many times the application polls the progress of a communication. The higher the value, the more CPU time is spent in polling.
mem_allocated_level	Gives the instantaneous memory usage by the library in bytes.
mem_allocated_highwater	Gives the maximum number of bytes ever allocated by the MPI library at a given process for the duration of the application.
coll_bcast_binom	Counts how many of the MPI broadcast collective calls use the Binomial algorithm during an application run.
num_shmem_coll	Counts how many of the collective communication calls are using shared memory.
coll_bcast_shmem	Counts how many of the MPI broadcast communication calls are shared memory based collectives.

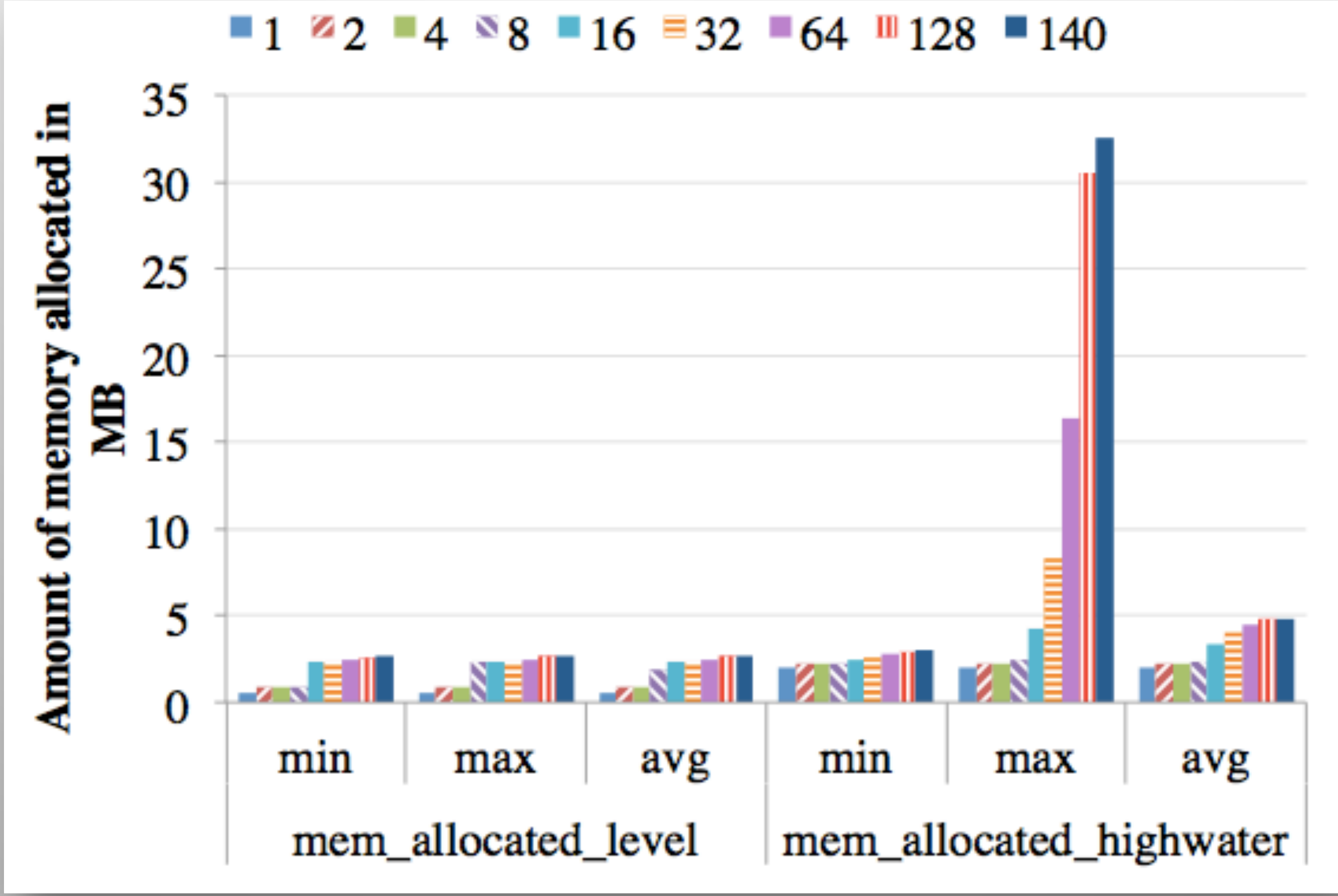
Case Study: Receive Queues for NAS BT



Case Study: Receive Queues for NEK5000



Case Study: Memory Consumptions for NEK5000



What Has to Happen Next?

- **Adoption across MPI implementation**
  - Availability of wide range of counters
  - Support of advanced mechanisms like configuration settings
  - Very good start by MPICH, MVAPICH and Open MPI
  - Integration of low level / device specific information
- **Tool support for counters**
  - Gyan is only a first step
    - Need more fine grained measurements
    - Need to look into correlations
  - Integration into other tools could be accelerated using a PAPI module
- **Common terminology for counters**
  - We won't get standardization, but we may get close
  - Would help consistency across tools

Other Tool Interfaces for Communication Analysis

- **More News from the MPI Tools Group … (planned)**
  - Tracing/Notification extensions to MPI_T
    - Ability to react to individual events
  - Attempt to revamp the PMPI interface
    - Callbacks instead of linker tricks
    - Ability to stack tools with clean interoperability
  - Join the discussion at **http://www.mpi-forum.org/** !!!
- **Need similar interfaces for other programming models**
  - Low level communication libraries
  - Higher level models with own communication (Charm++, Legion)
  - Low level communication libraries
  - Hardware support (?)
  - ➢ Must be baked in from the beginning

Conclusions

- **Modern networks pose challenges for the end user**
  - Node placement, variability, need to pinpoint contention
  - Will be even worse at exascale
  - Need introspection into network performance
- **Initial approaches**
  - Topology aware visualization of existing counters
  - Simulation infrastructures
- **Need more introspection into actual networks and their performance**
- **Standardized interfaces are key**
  - Enable tool portability
  - Enable clean correlation
  - Should be part of any message-based library or programming model

The Scalability Team

http://scalability.llnl.gov/

Abhinav Bhatele | David Boehme | Todd Gamblin | Tanzima Islam | Ignacio Laguna | Kathryn Mohror | Barry Rountree | Martin Schulz

- **Main topics**
  - **Performance analysis tools and optimization**
  - Correctness and debugging (incl. STAT, AutomaDeD, MUST)
  - Tool infrastructures (incl. P^{n}MPI, GREMLINs)
  - Power-aware and power-limited computing (incl. Adagio)
  - Resilience and Checkpoint/Restart (incl. SCR)
- **Funding sources involved in presented work:**

Conclusions

http://scalability.llnl.gov/

- **Modern networks pose challenges for the end user**
  - Node placement, variability, need to pinpoint contention
  - Will be even worse at exascale
  - Need introspection into network performance
- **Initial approaches**
  - Topology aware visualization of existing counters
  - Simulation infrastructures
- **Need more introspection into actual networks and their performance**
- **Standardized interfaces are key**
  - Enable tool portability
  - Enable clean correlation
  - Should be part of any message-based library or programming model