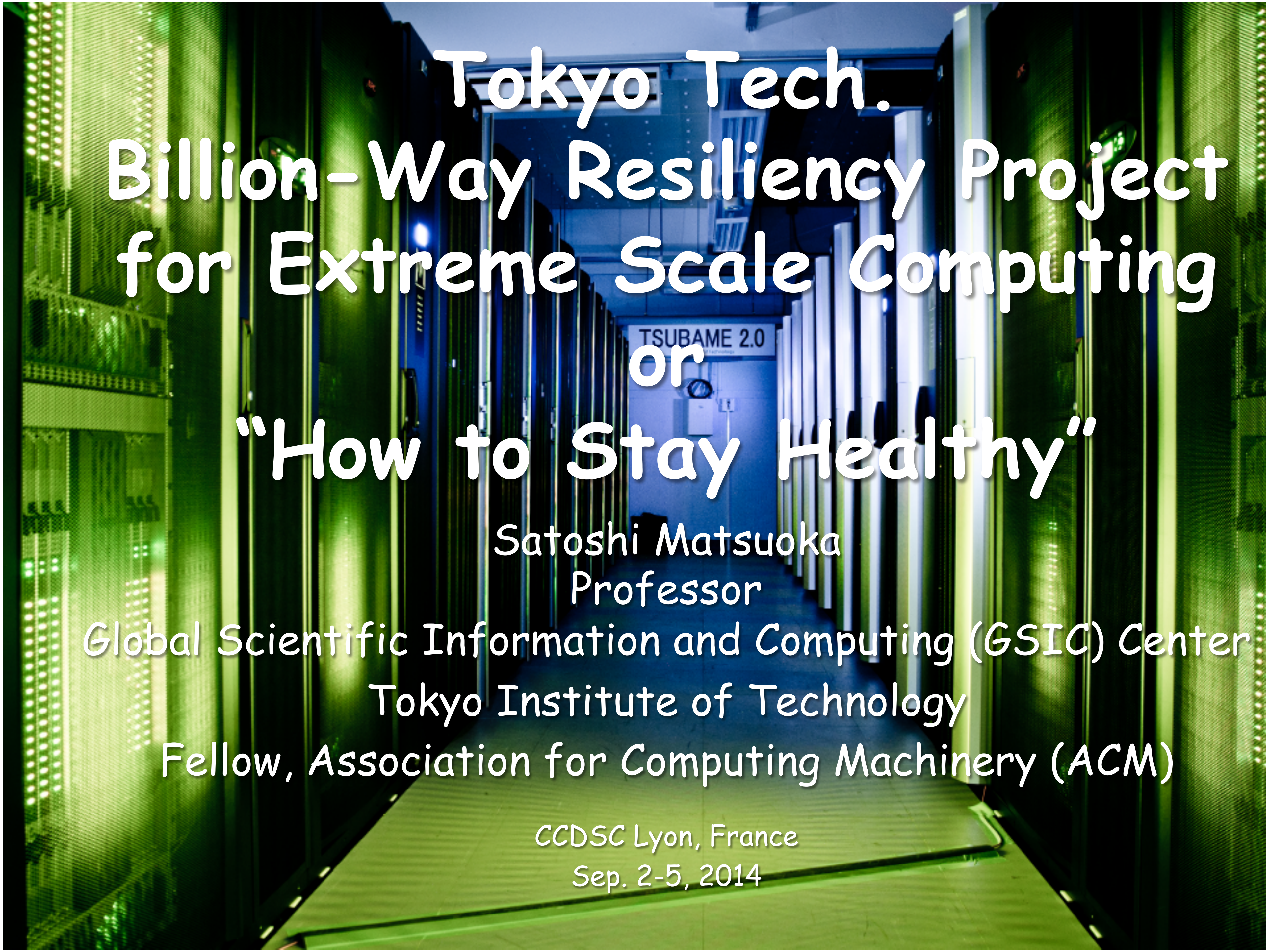


Tokyo Tech. Billion-Way Resiliency Project for Extreme Scale Computing or "How to Stay Healthy"

Satoshi Matsuoka
Professor
Global Scientific Information and Computing (GSIC) Center
Tokyo Institute of Technology
Fellow, Association for Computing Machinery (ACM)

CCDSC Lyon, France
Sep. 2-5, 2014

Tokyo Tech. Billion-Way Resiliency Project (2011-2015)

- Collaboration with ANL (Franck Cappello, FTI), LLNL (Bronis de Spinksi, SCR), Hideyuki Jitsumoto (U-Tokyo)…
- More precise system fault model and associated cost model of recovery and optimization
- Aggressive architectural, systems, and algorithmic improvements
  - Use of localized flash/NVM for ultra fast checkpoints and recovery
  - Advanced coding and clustering algorithms for reliability against multiple failures
  - Combining coordinated & uncoordinated checkpoints
  - Overlapping transfers in the checkpoint storage hierarchy for quick recovery
  - Power optimized checkpoints
- Better monitoring and micro-recovery

理化学研究所 RIKEN

東京大学 THE UNIVERSITY OF TOKYO

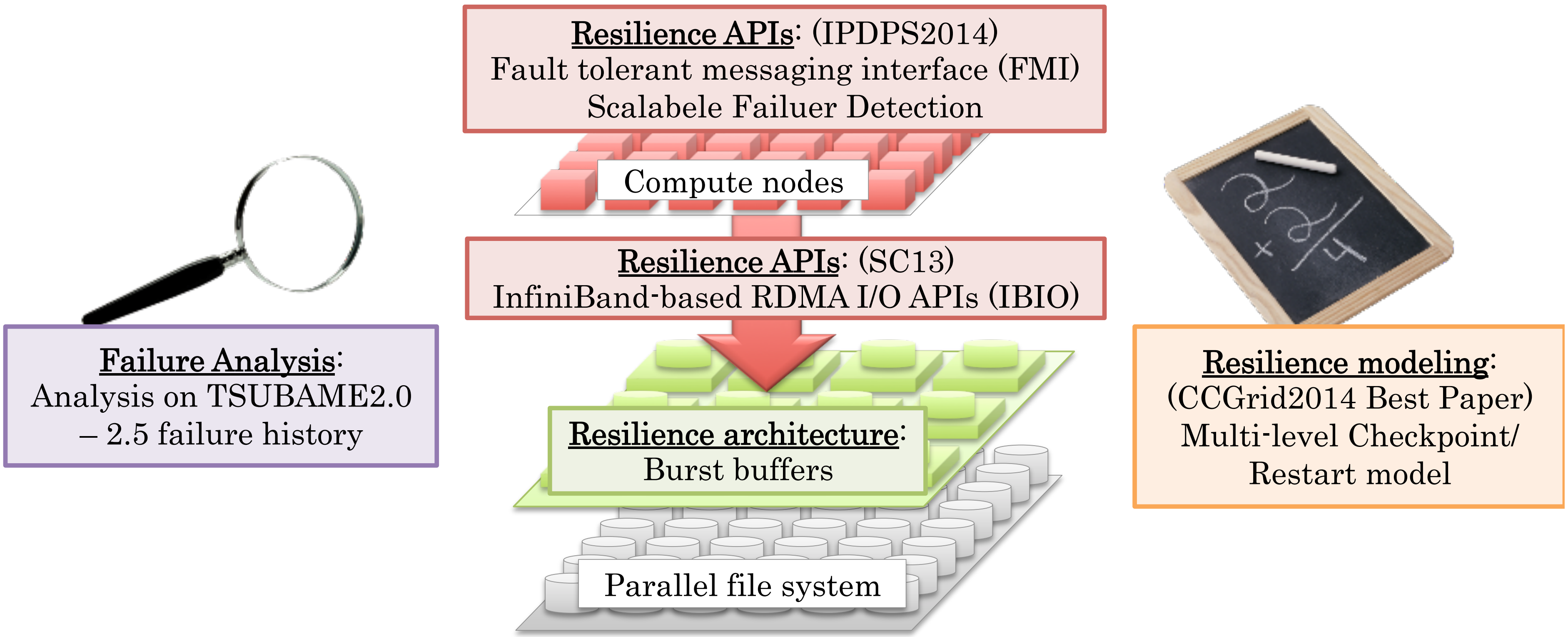
TOKYO TECH Pursuing Excellence

Inria INVENTEURS DU MONDE NUMÉRIQUE

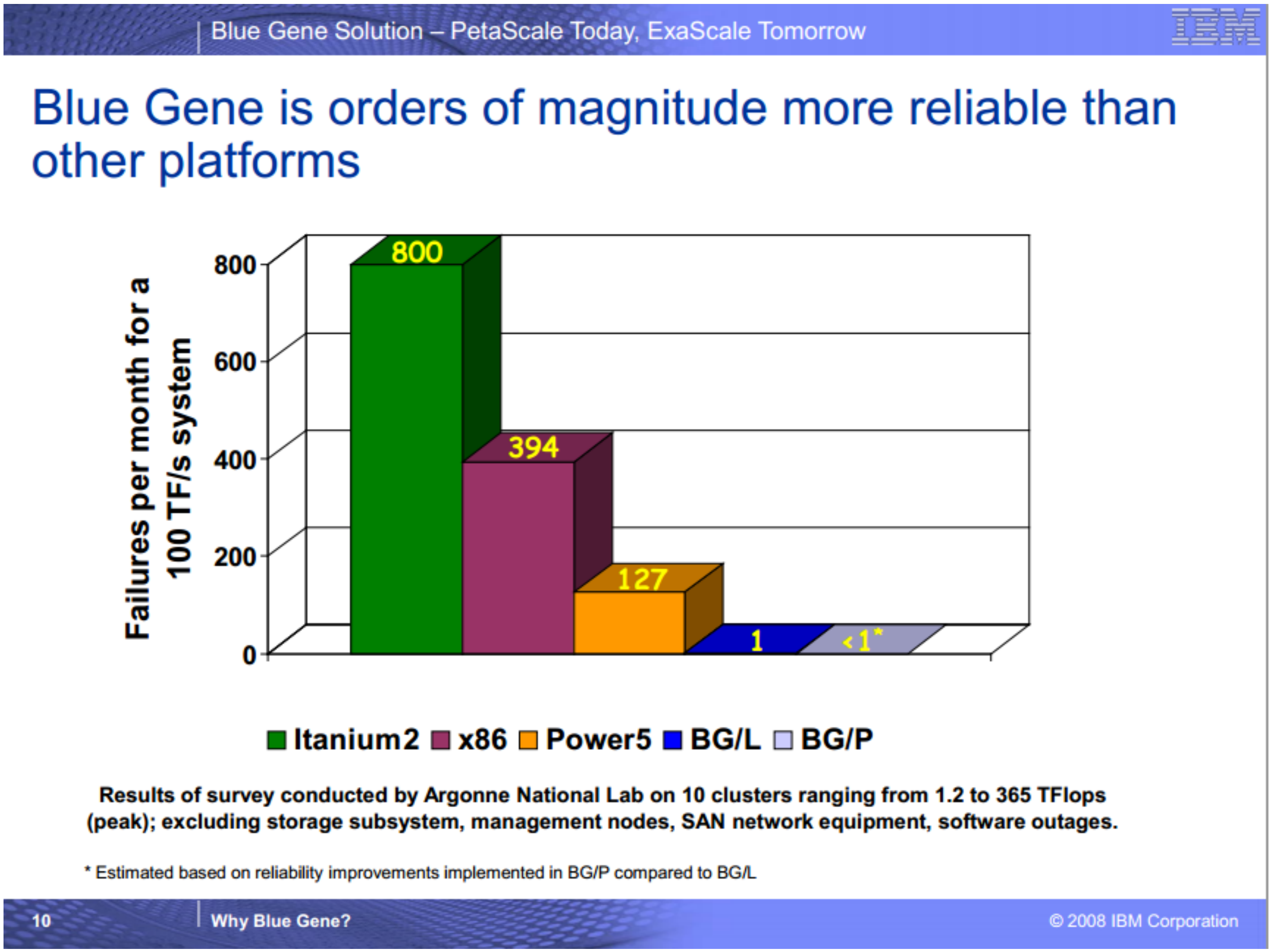
Lawrence Livermore National Laboratory

Holistic Approach to Billion-way Resiliency

- Failure Analysis, APIs, Modeling and Architectures driving multi-level checkpoint/restart through extensive collaborations between LLNL and Tokyo Tech



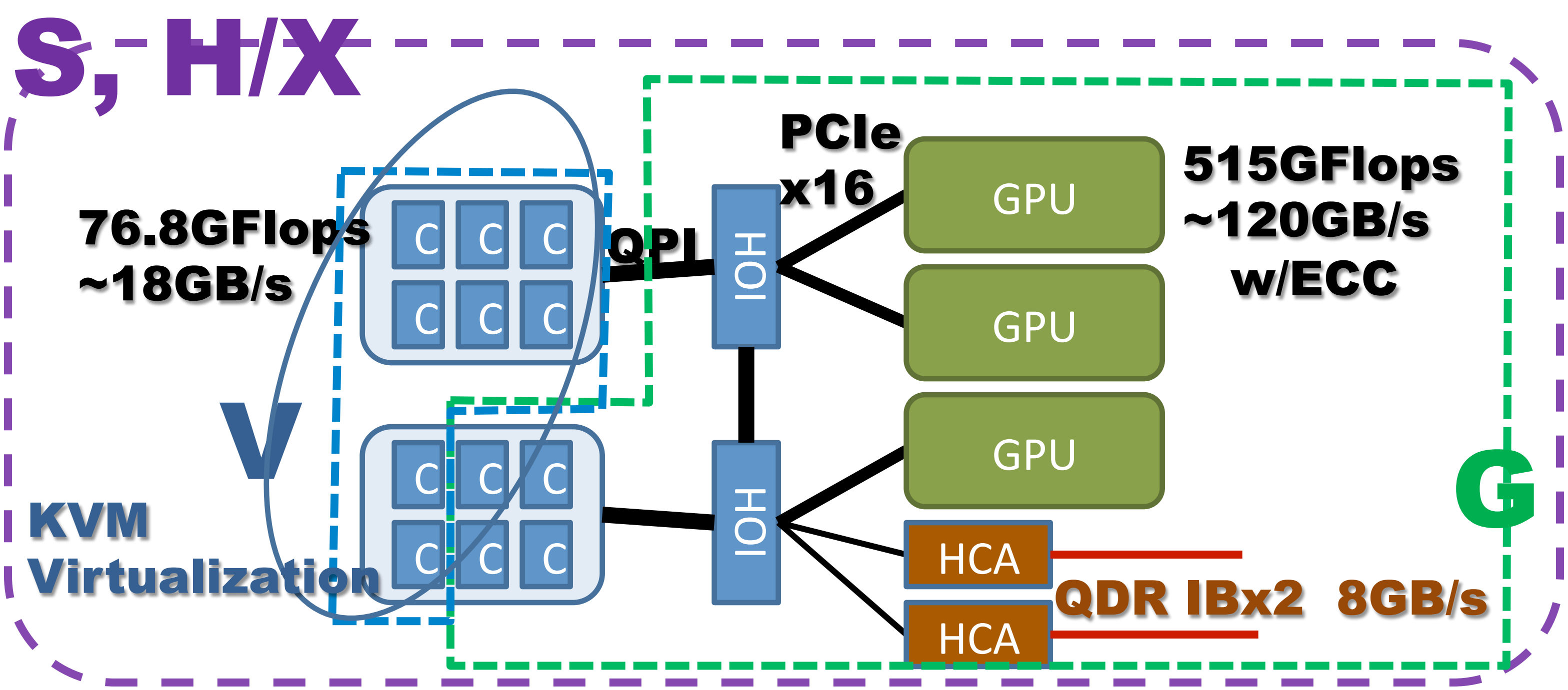
A cluster-based SC like TSUBAME is not supposed to work…



Service List

service	assigned nodes		running jobs		users
S	100%	352 / 352 nodes	38%	175 / 452 jobs	46
S96	100%	41 / 41 nodes	53%	41 / 76 jobs	4
G	99%	475 / 477 nodes	100%	62 / 62 jobs	12
V	83%	364 / 437 nodes	80%	1531 / 1904 jobs	34
L128	100%	10 / 10 nodes	66%	10 / 15 jobs	1
L128F	100%	10 / 10 nodes	71%	10 / 14 jobs	1
L256	37%	3 / 8 nodes	[illegible]	3 / 3 jobs	1
L512	100%	2 / 2 nodes	100%	2 / 2 jobs	1
H/X	93%	301 (+ 95) / 420 nodes	100%	87 / 87 jobs	8
ALL	93%	1558 (+ 95) / 1757 nodes	73%	1921 / 2615 jobs	93

~2000 SC Users
~93% System Utilization
~50% GPU Utilization



Why Does TSUBAME Work?

- For the many-core era, component complexity / Flops do not differ tremendously across machines
- Thus, hard-stop component failures will occur fairly equally
  - But may not lead to application faults if detected early
- Many application errors also attributed to system software's inability to scale with reliable operations, especially with domino effects
  - Race conditions leading to anomalous pauses which will screw up your deamons which in turn de-mounts your file system which in turn…

Every fault is recorded and made immediately public

mon.g.gsic.titech.ac.jp/trouble-list/index.htm

計算サ… メインカードの変更 SC 法人入会 YouTube トップ おすすめサイト Tag Heuer Monac… 2009-11-22 - なべ… microsecond

TSUBAME2.0 障害履歴[Failure History of TSUBAME2.0]

last update : 2013.06.18

1.障害発生情報[Current Trouble Information of TSUBAME2.0]

ホスト/機器	キュー	発生日付	復旧日付	障害状況	原因	対処	影響範囲	カテゴリ
t2a001052	S	2013/06/17 11:22	2013/06/17 17:00	Uncorrectable PCI Express Error	GPU障害	GPU2抜き差し	該当ノード	GPU
t2a000059	S	2013/06/17 00:22	2013/06/17 17:00	Uncorrectable PCI Express Error	GPU障害	GPU0抜き差し	該当ノード	GPU
t2a004111	HX	2013/06/16 13:33	2013/06/18 17:10	Uncorrectable PCI Express Error Uncorrectable Machine Check Exception	CPU障害	CPU1交換	該当ノード	CPU
t2a001058	S	2013/06/15 23:22	2013/06/17 17:00	Uncorrectable PCI Express Error	GPU障害	GPU0抜き差し	該当ノード	GPU
t2a010065	PoolS	2013/06/15 16:00-17:00	2013/06/17 09:30	負荷高騰	-	再起動	該当ノード	OtherSW
t2a000057	S	2013/06/15 10:35-11:35	2013/06/17 09:30	ssh不可	-	ssh再起動	該当ノード	OtherSW
t2a001054	S	2013/06/15 22:35-23:35	2013/06/17 09:30	ssh不可	-	ssh再起動	該当ノード	OtherSW
t2a001125-vm1 t2a001127-vm1 t2a001130-vm1 t2a001169-vm1 t2a002094-vm1		2013/06/10-2013/06/14	2013/06/14	仮想ノードダウン	調査中	仮想マシン再起動	該当ノード	
t2a000010 t2a000012 t2a000044 t2a000169	S	2013/06/14	2013/06/17	Uncorrectable PCI Express Error	GPU障害	GPU2抜き差し	該当ノード	GPU
t2a000082	S	2013/06/14	2013/06/17	Uncorrectable PCI Express Error	GPU障害	GPU0抜き差し	該当ノード	GPU
t2a004115	HX	2013/06/14	2013/06/18	Uncorrectable PCI Express Error	GPU障害	GPU1,2スワップ	該当ノード	GPU
t2a000030	S	2013/06/14	2013/06/14	ssh不可	-	再起動	該当ノード	OtherSW
t2a000072 t2a000117	S	2013/06/13	2013/06/13	Uncorrectable PCI Express Error	GPU障害	GPU2抜き差し	該当ノード	GPU
t2a000078	S	2013/06/13	2013/06/13	Uncorrectable PCI Express Error	GPU障害	GPU0抜き差し	該当ノード	GPU
t2a001024	S	2013/06/13	2013/06/13	Uncorrectable PCI Express Error	GPU障害	GPU1,2スワップ	該当ノード	GPU
t2a001160	GV	2013/06/11	2013/06/11	Uncorrectable PCI Express Error	GPU障害	GPU1,2スワップ	該当ノード	GPU
t2a002027	GV	2013/06/11	2013/06/11	Uncorrectable PCI Express Error	GPU障害	GPU2抜き差し	該当ノード	GPU
t2a001028	S	2013/06/10	2013/06/10	Uncorrectable PCI Express Error	GPU障害	GPU2抜き差し	該当ノード	GPU
t2a002178	GV	2013/06/10	2013/06/10	Uncorrectable PCI Express Error	GPU障害	GPU1,2スワップ	該当ノード	GPU
t2a003129	GV	2013/06/10	2013/06/10	Uncorrectable PCI Express Error	GPU障害	GPU0抜き差し	該当ノード	GPU
t2a004172	HX	2013/06/10	2013/06/10	Uncorrectable PCI Express Error	GPU障害	GPU0,1スワップ	該当ノード	GPU
t2a006025 t2a006026 t2a006027 t2a006028	HX	2013/06/10	2013/06/10	System Power Supply: General Failure	PSU障害	P/S Bay2交換	該当ノード	PSU
t2a001134-vm1 t2a001158-vm1 t2a001174-vm1 t2a002013-vm1 t2a002019-vm1 t2a002028-vm1 t2a002074-vm1 t2a002084-vm1 t2a003017-vm1		2013/06/03-2013/06/07	2013/06/07	仮想ノードダウン	調査中	仮想マシン再起動	該当ノード	
MSC10418		2013/06/07	2013/06/10	Moisture Error	ラック障害	Condensation pomp& Water	ラック内の ノード	OtherHW

Log Sanitization Process

- Obvious erroneous entries in error log
  - SSD failure categorized as "GPU failure"
  - Simple "node down" vs. "CPU failure and replace"
- Initial failures in the "bathtub curve" misleading
  - TSUBAME2.0 commissioned Nov. 2010
  - Stable year period Aug.1 2012 to July 31, 2013
- Missing info in error log
  - No indication of
  - extrapolation of effect of failures
- Anomalous, very specific failures caused by unresolved "bug" in HW (see next slide)

Yearly Distribution of Faults in TSUBAME2.0

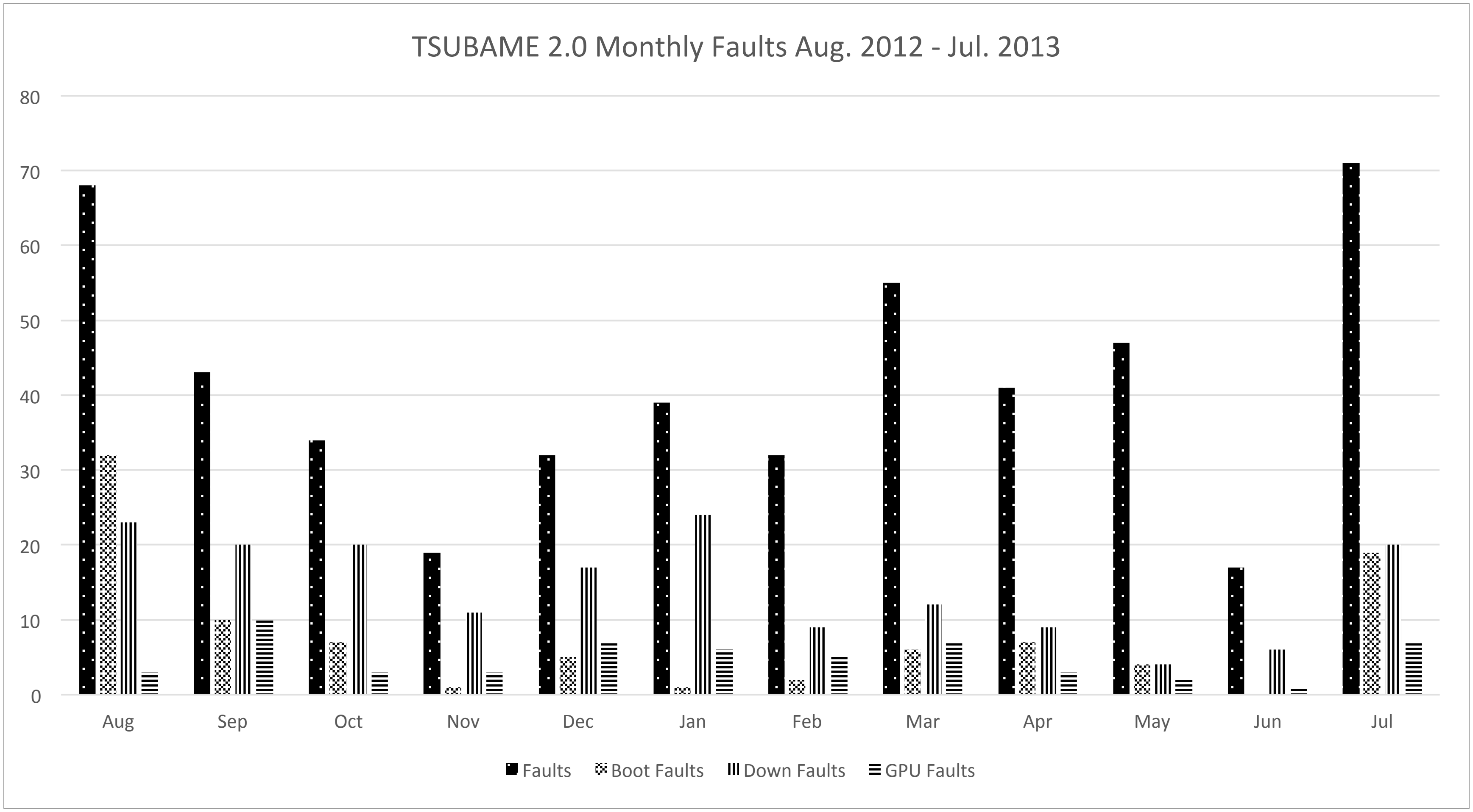


Table 1 TSUBAME2.0
Aug 1 2012 ~ July 31, 2013Failure analysis of components

	SUM	Boot Failures	Fail-stop Failures	Unaffected	Multi Failures	Repairs	GPU Repairs	DIMM Repairs
Unknown Boot Failure	35	35				0		
CPU	16	4	12			14		
Disk/Storage (wo SSD)	17	5		12		5		
Unknown Node Failure	15		15			0		
Fan	100			100		13		
GPUs								
GPU-PCI	362		362			96	39	
GPU-Link	16			16		10	3	
GPU-ECC	10		10			10	10	
GPU-Unknown	10	6	4			9	5	
Memory	12	1	7	4		11		14
Network								
Infiniband	22		4	16	2	4		
Other Networks	78		55	23		0		
Other HW	27	22		5		24		
Batch System (PBS Pro)	13		3	10		0		
PSU	33		26	7		33		
Rack	6			5	1	4		
SSD	34	12	22			32		
System Board	22	9	13			22		
Total	828	94	533	198	3	287	57	14
Corrected Total	498	94	210	191	3	209	57	14

Overview of Analysis (1)

- **TSUBAME2.0 highly reliable**
  - 500 failures, only 210 fail stop / year
    - System MTTI = 1.7 days, node MTTI = 2500 days
    - Much better than conjectured MTTI of K computer
- **GPU comparatively reliable vs. CPUs**
  - 19 CPU+memory fail-stop failures, 25 replacements, MTBF 118 years, 2.22^{18} FLOP/error
  - 53 GPU+memory ECC fail-stop failures, 57 replacements, MTBF 75 years, 1.61^{19} FLOP/error
  - GPU error rate x7 better / flop vs. CPU, proportional to performance difference per chip
- CPU+GPU 7216 units: if chip-level MTBF is similar for TSUBAME3.0, 25-30 Petaflop possible in 2015-16

Overview of Analysis (2)

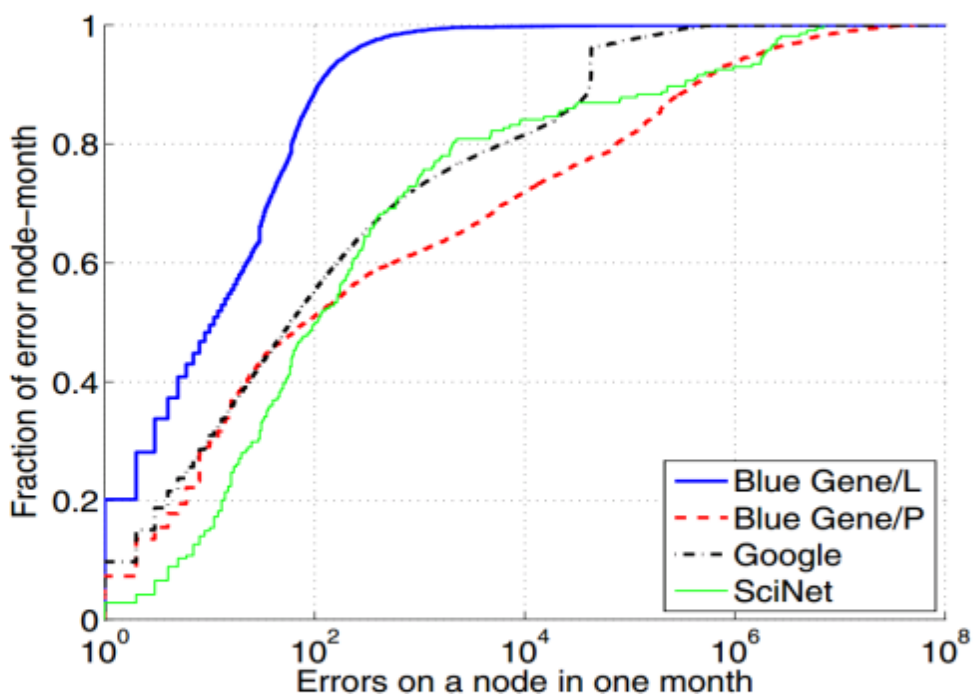
- **Failures are Largely Independent**
  - Only 3 multi-node failures out of 210 fail stops
  - Low # of Infiniband and storage failures
- **TSUBAME2.0 Fat Node architecture vs. C.f. Many Thin-nodes architecture e.g. BG/Q**
  - Most failures contained within nodes
    - C.f. BG/Q – failure in node compromises the entire task → parameter sweep jobs NG
  - Local checkpoint & dynamic recovery very effective

The reality speaks… DRAM Error Rates

- Andy A. Hwang, Ioan Stefanovici, and Bianca Schroeder. "Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications

System	Time (days)	Nodes	# DIMMs	DRAM in system (TB)	TByte years	Nodes with errors	Nodes w/ chipkill errs	Total # Errors	FIT
BG/L	214	32,768	N/A	49	28	1,742 (5.32%)	N/A	$227 \cdot 10^6$	97,614
BG/P	583	40,960	N/A	80	127	1,455 (3.55%)	1.34%	$1.96 \cdot 10^9$	167,066
SciNet	211	3,863	31,000	62	35	97 (2.51%)	N/A	$49.3 \cdot 10^6$	18,825
Google	155	20,000	~ 130,000	220	93	20,000	N/A	$27.27 \cdot 10^9$	N/A

Table 1. Summary of system configurations and high-level error statistics recorded in different systems



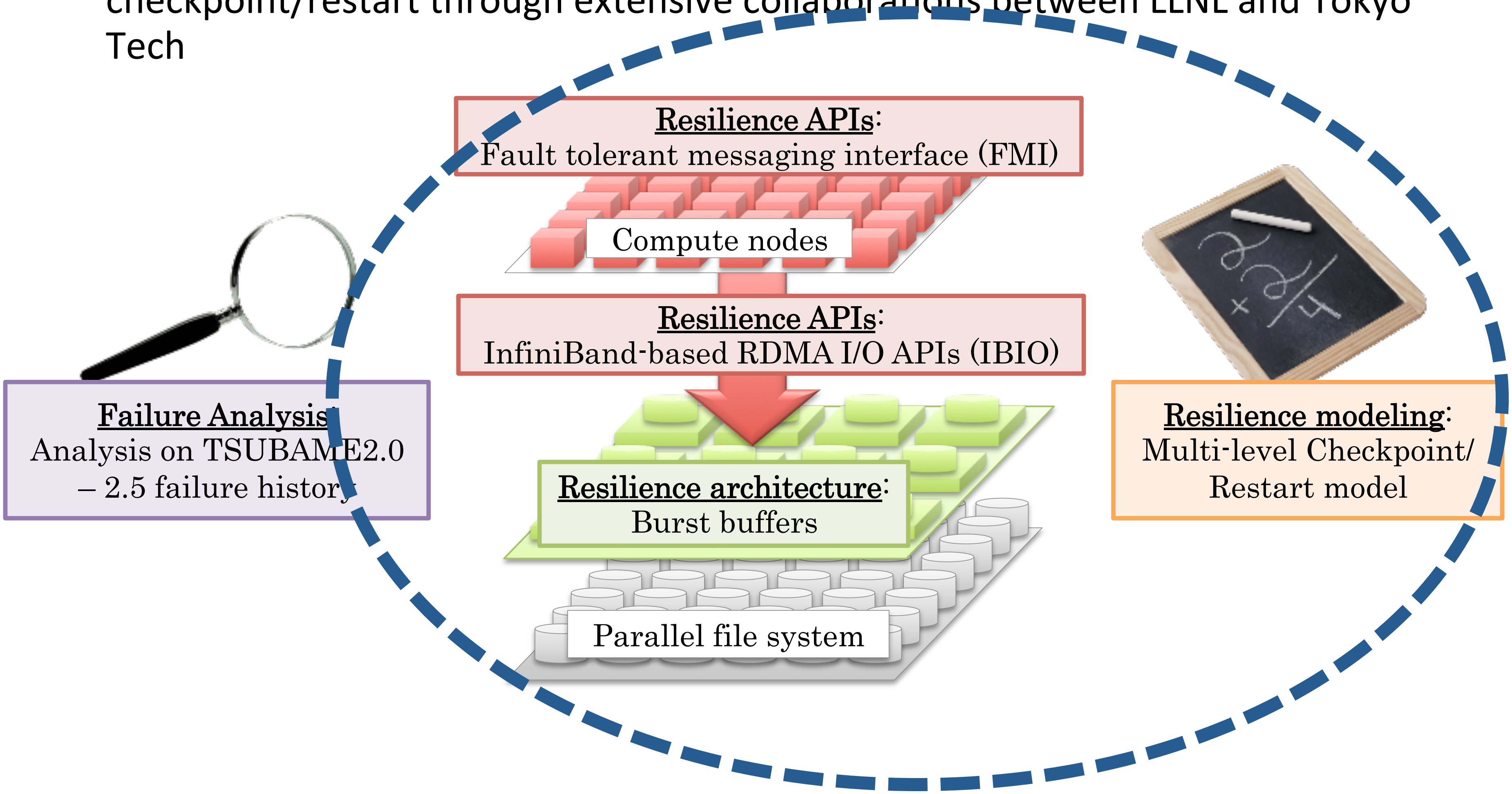
No significant difference in node DRAM error rates (in fact significantly worse for corrected errors for BG)

Overview of Analysis (3)

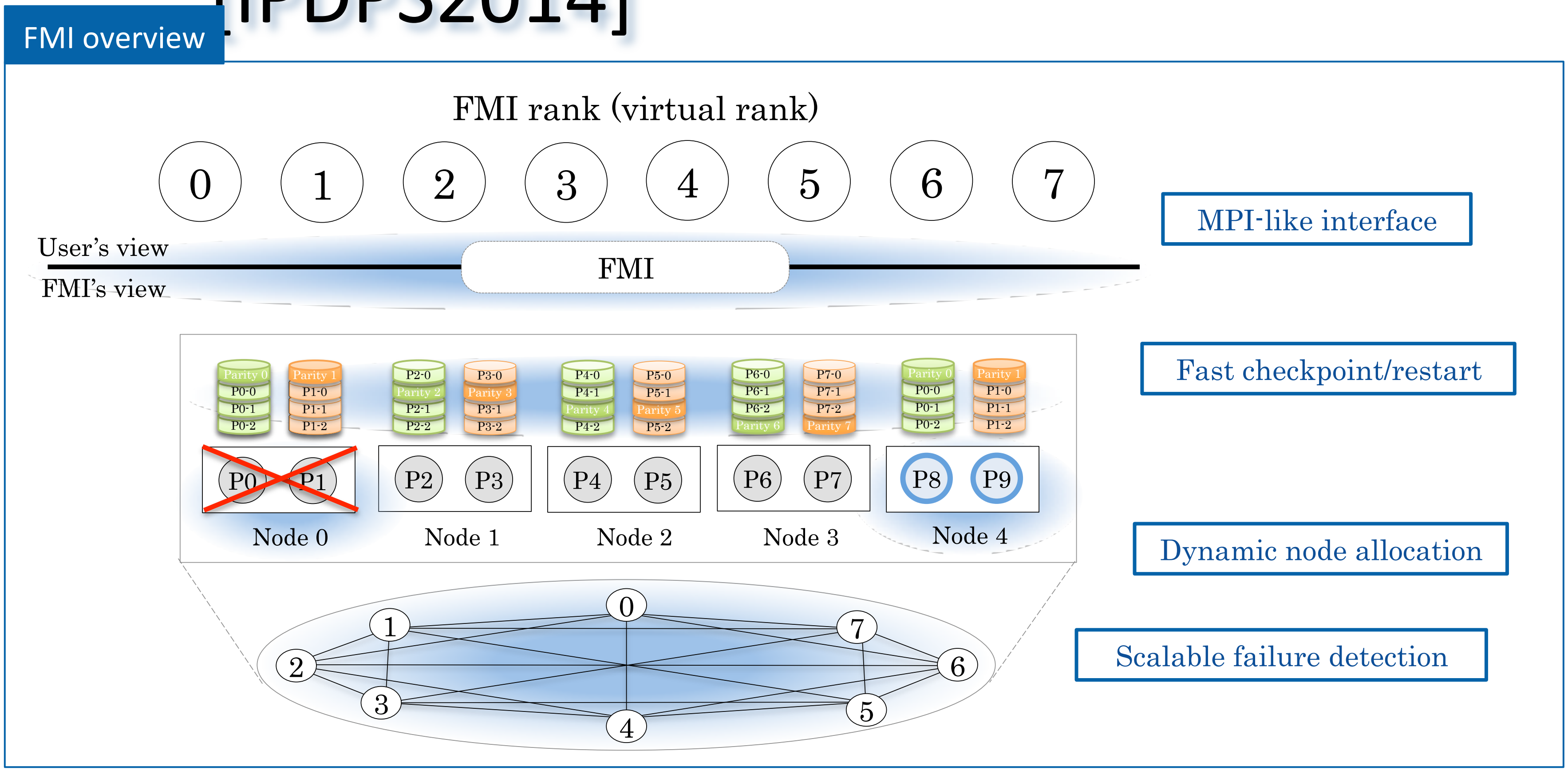
- **<u>Memory failures consistent or better c.f. previous work</u>**
  - 17,000 DIMMs an 4264 GPUs in TSUBAME2.0
  - 14 DIMM DUE errors, 10 GPU double bit EC Errors / Year
  - DUE DIMM errors 0.082% vs. Google[8] 0.22%
  - GPU memory error 0.23% vs. 0.83% BG/P Chipkill [10]
  - K Computer 700,000 DIMMs => 600 DIMM failures predicted with same error rate as TSUBAME2.0 => **<u>MTBF ~=1/2 day</u>**
- Failures seasonal, but not due to temperature
  - Largely due to boot failures in peak-shift operations during summer to limit power, despite SW retries
  - Future SCs in Clouds need to cope with this

Holistic Approach to Billion-way Resiliency (Modeling and Permeation thru Software Stack)

- Failure Analysis, APIs, Modeling and Architectures driving multi-level checkpoint/restart through extensive collaborations between LLNL and Tokyo Tech



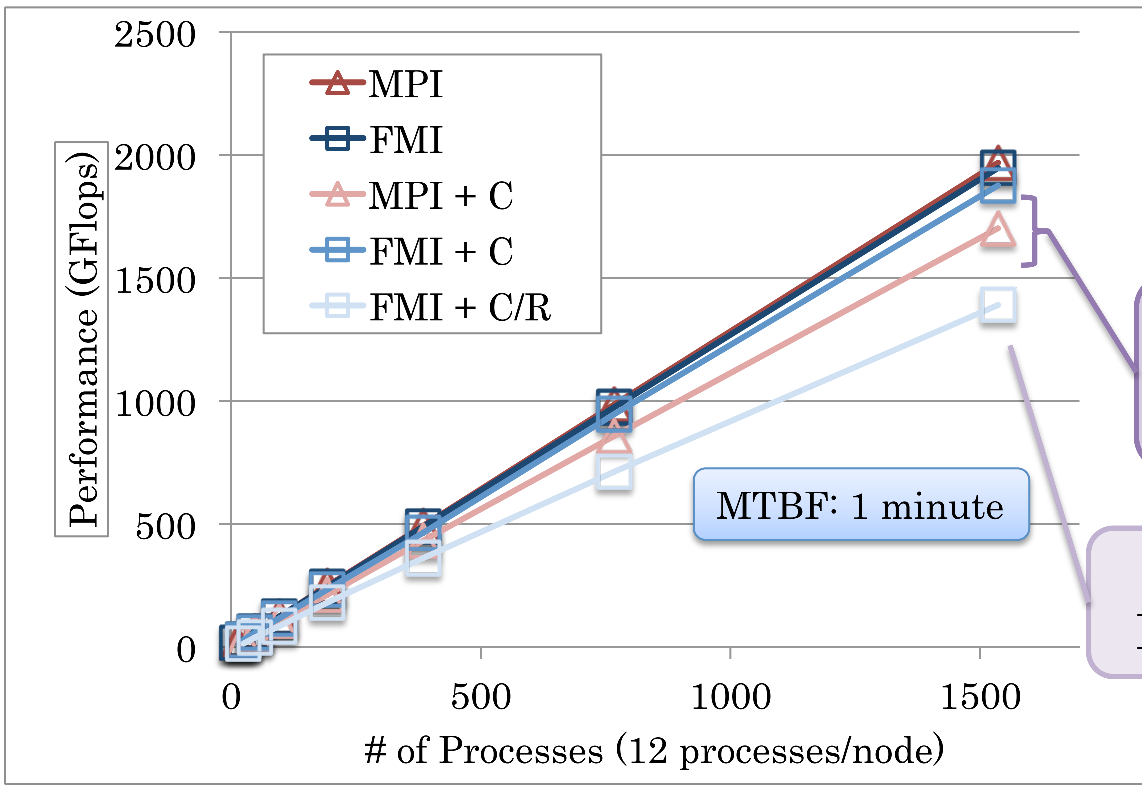
FMI: Fault Tolerant Messaging Interface [IPDPS2014]



- FMI is a survivable messaging interface providing MPI-like interface
  - Scalable failure detection ⇒ Overlay network
  - Dynamic node allocation ⇒ FMI ranks are virtualized
  - Fast checkpoint/restart ⇒ Diskless checkpoint/restart

Application runtime with failures

- Benchmark: Poisson's equation solver using Jacobi iteration method
  - Stencil application benchmark
  - MPI_Isend, MPI_Irecv, MPI_Wait and MPI_Allreduce within a single iteration
- For MPI, we use the SCR library for checkpointing
  - Since MPI is not survivable messaging interface, we write checkpoint memory on tmpfs
- Checkpoint interval is optimized by Vaidya's model for FMI and MPI



P2P communication performance

	1-byte Latency	Bandwidth (8MB)
MPI	3.555 usec	3.227 GB/s
FMI	3.573 usec	3.211 GB/s

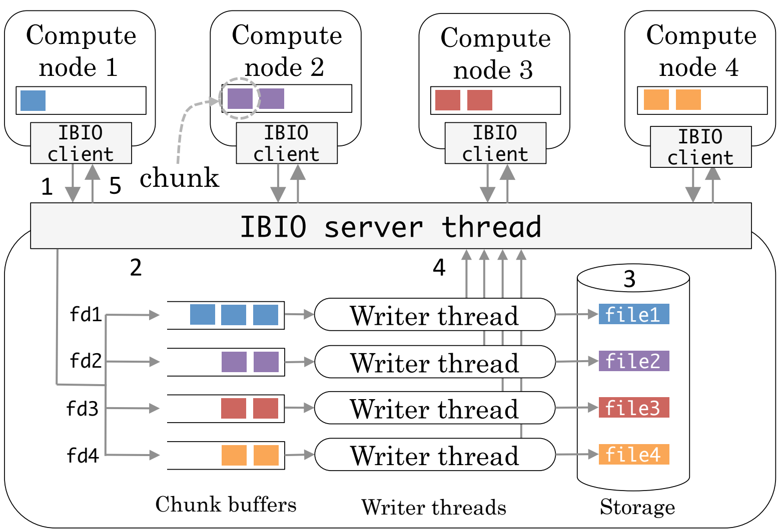
FMI directly writes checkpoints via memcpy, and can exploit the bandwidth

Even with the high failure rate, FMI incurs only a 28% overhead

APIs for burst buffers:[SC13]
InfiniBand-based I/O interface (IBIO)

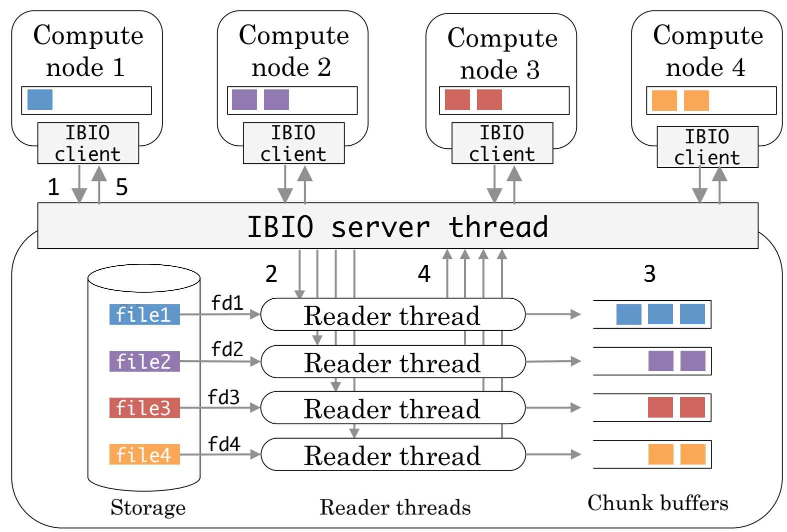
- Provide POSIX I/O interfaces
  - open, read, write and close
  - Client can open any files on any servers
    - `open("hostname:/path/to/file", mode)`
- IBIO use ibverbs for communication between clients and servers
  - Exploit network bandwidth of infiniBand

IBIO write



IBIO write: four IBIO clients and one IBIO server

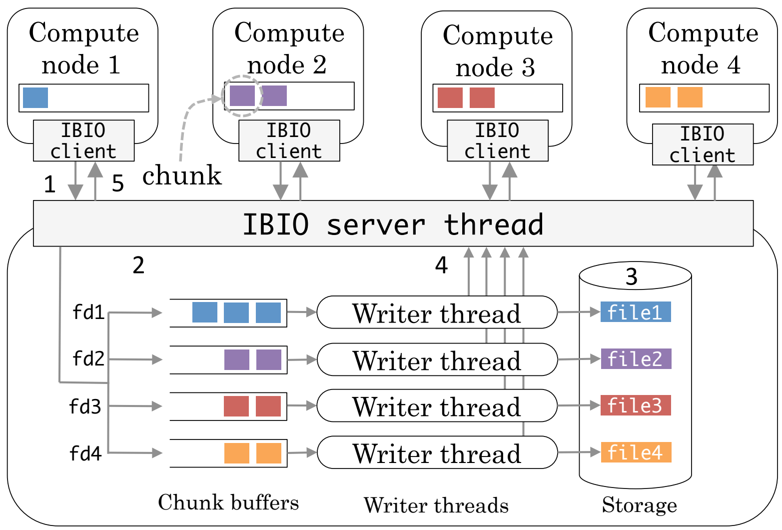
IBIO read



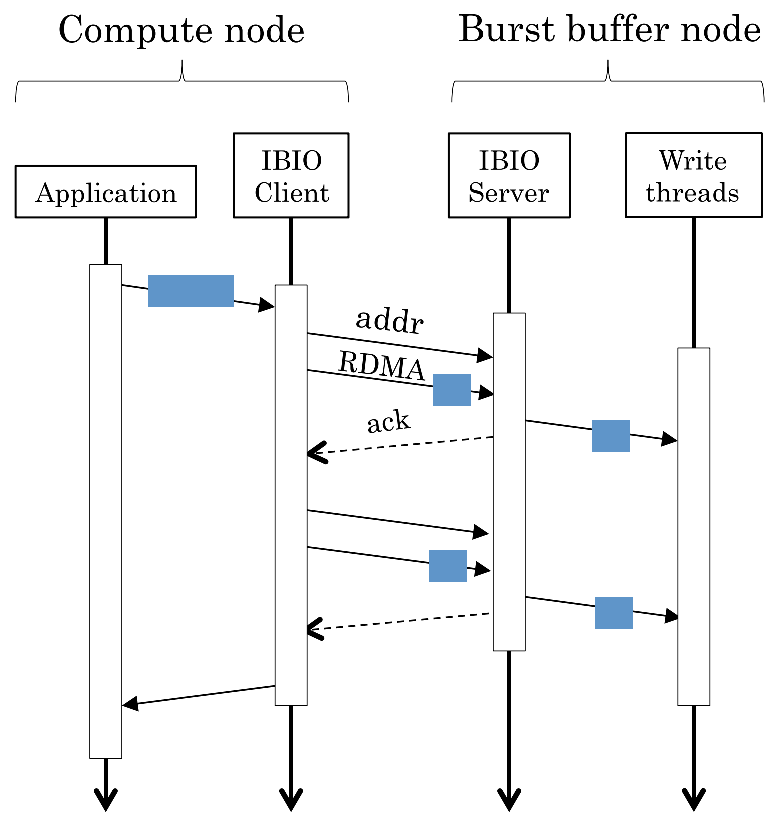
IBIO read: four IBIO clients and one IBIO server

IBIO write/read

IBIO write



IBIO write: four IBIO clients and one IBIO server



- IBIO write
  1. Application call IBIO client function with data to write
  2. IBIO client divides the data into chunks, then send the address to IBIO server for RDMA
  3. IBIO server issues RDMA read to the address, and reply ack
  4. Continues until all chunks are sent, and return to application
  5. Writer threads asynchronously write received data to storage
- IBIO read
  – Reads chunks by reader threads and send to clients in the same way as IBIO write by using RDMA

Resilience modeling overview [CCGrid2014 Best Paper]

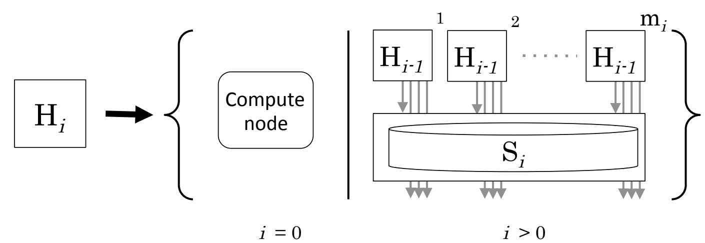
- To find out the best checkpoint/restart strategy for systems with burst buffers, we model checkpointing strategies

C/R strategy model

$$O_i = \begin{cases} C_i + E_i & \text{(Sync.)} \\ I_i & \text{(Async.)} \end{cases} \qquad L_i = C_i + E_i$$

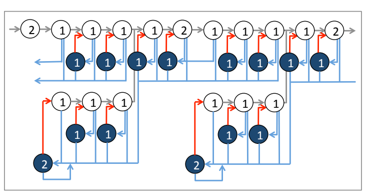
$$C_i \text{ or } R_i = \frac{\text{< C/R date size / node >} \times \text{<\# of C/R nodes per } S_i^* \text{ >}}{\text{< write perf. (} w_i \text{) > or <read perf. (} r_i \text{) >}}$$

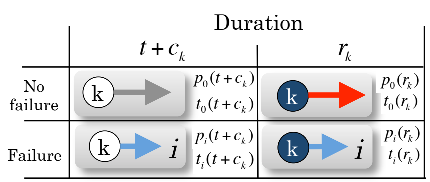
Recursive structured storage model



Storage Model: $H_N \{m_1, m_2, \ldots, m_N\}$

MLC model [2]





t : Interval
c_c : c-level checkpoint time
r_c : c-level recovery time
λ_i : i-level checkpoint time

$p_0(T) = e^{-\lambda T}$
$t_0(T) = T$
$p_i(T) = \frac{\lambda_i}{\lambda}(1 - e^{-\lambda T})$
$t_i(T) = \frac{1 - (\lambda T + 1) \cdot e^{-\lambda T}}{\lambda \cdot (1 - e^{-\lambda T})}$

$p_0(T)$: No failure for T seconds
$t_0(T)$: Expected time when $p_0(T)$
$p_i(T)$: i - level failure for T seconds
$t_i(T)$: Expected time when $p_i(T)$

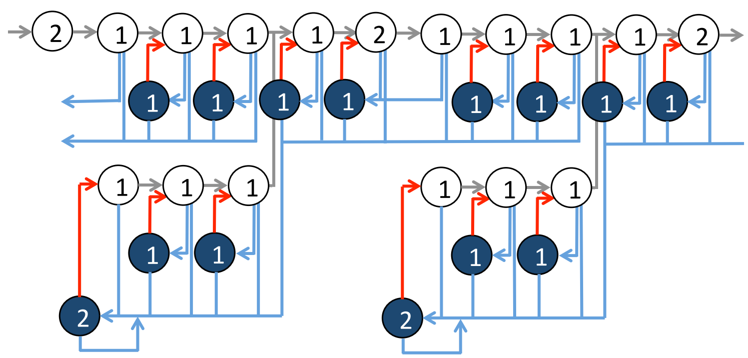
Efficiency

Fraction of time an application spends only in useful computation

[2] Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Design and Modeling of a Non-blocking Checkpointing System", SC12

Resilience modeling:
Multi-level Checkpoint/Restart model

Asynchronous checkpointing model



t : Interval

c_c : c-level checkpoint time

r_c : c-level recovery time

	Duration $t + c_k$		r_k	
No failure	k →	$p_0(t+c_k)$ $t_0(t+c_k)$	k →	$p_0(r_k)$ $t_0(r_k)$
Failure	k → i	$p_i(t+c_k)$ $t_i(t+c_k)$	k → i	$p_i(r_k)$ $t_i(r_k)$

$p_0(T)$: No failure for T seconds

$t_0(T)$: Expected time when $p_0(T)$

$p_i(T)$: i - level failure for T seconds

$t_i(T)$: Expected time when $p_i(T)$

Poisson's distribution

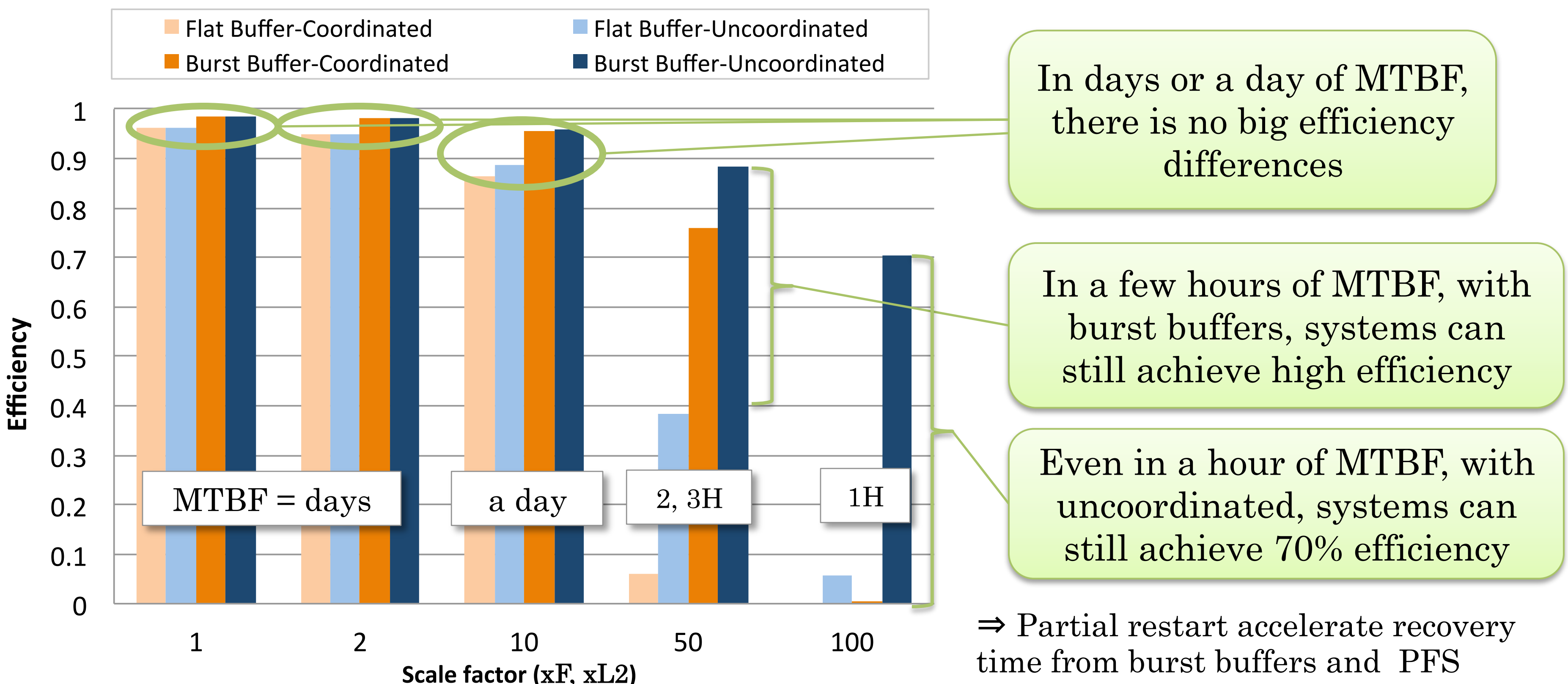
$$p_0(T) = e^{-\lambda T}$$
$$t_0(T) = T$$
$$p_i(T) = \frac{\lambda_i}{\lambda}(1 - e^{-\lambda T})$$
$$t_i(T) = \frac{1 - (\lambda T + 1) \cdot e^{-\lambda T}}{\lambda \cdot (1 - e^{-\lambda T})}$$

λ_i : i-level checkpoint time

$$\lambda = \sum \lambda_i$$

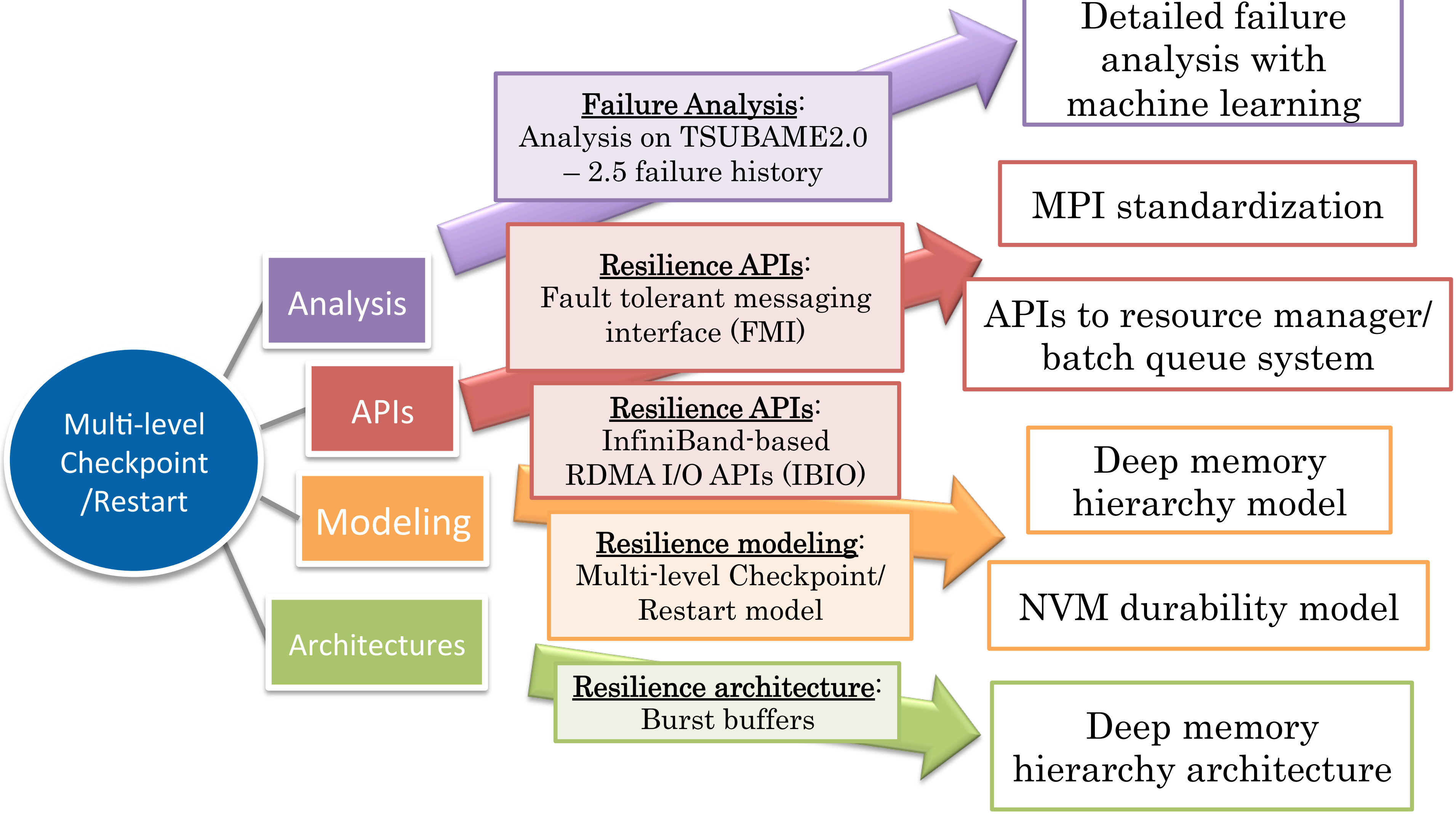
Efficiency with Increasing Failure Rates and Checkpoint Costs

- Assuming there is no message logging overhead



⇒ Partial restart accelerate recovery time from burst buffers and PFS checkpoint

Near term work



However, we are not there yet

- How do we proactively prevent faults, and assume such correction in the overall model and sys software?
- How do we detect "faults?"
  - Some advances fault injection / ABFT-style fault detection
  - However, real machine failure modes are extremely elusive
    - We face these every day with TSUBAME…
  - How do we distinguish between application bugs, system software bugs, "ephemeral" soft errors, moderately failing hardware, and hard error crashes?
- What is the right recovery for each failure mode?
  - "Recover node state and try again" only partially applicable to tremendously abundant set of failure modes
  - There are various algorithms but they need to scale to 100,000 nodes or more…

TSUBAME2.0 Periodic Health Check List

Check Category	Check Performed	Interval	Action on Fautl	Subject	Av. Exec Time
Network	Infiniband Status, Check	2H	Notify Sysadmin	Node	5.6E-02
Clock	System Clock Drift	2H	Notify Sysadmin	Node	2.4E-01
GPU	PCIe Link Speed, Driver Permission, Device Memory ECC Error	2H	Auto Offline	Node	7.8E-02
HDD	Available Space, Filesystem Mount	2H	Notify Sysadmin	Node	1.2E-02
SSD	Partition and Size	2H	Notify Sysadmin	Node	Ditto
SSD	Permission	1D		Node	Ditto
SSD	fsck /scratch space	1H	Notify Sysadmin	Node	Ditto
SSH	SSH login deamon	1H	Auto Offline	All Nodes	2.3E+01
Process	Zombie Process	1H	Kill Zombie	Node	6.2E+00
PBS	PBS scheduler status, qstat response (60 seconds)	1H	Notify Sysadmin	Admin Node	2.3E-01
PBS	MOM Check			Node	5.4E-02
PBS	Decommision Waiting Reserve Job	1H	Auto Decommisioning	Admin Node	6.5E+00
OpenSM	Check operation	1H	Notify Sysadmin	Admin Node	7.7E+01
Lustre	Check MDS, OSS, OST activity	1H	Notify Sysadmin	Admin Node	1.5E-01
Interactive	Load Average	1D	Notify Sysadmin	Interactive	8.0E-03
H (Reservation) Queue	Check Actual reservation and batch status	1D	Notify Sysadmin	Admin Node	4.4E-01
VM Check	SSH Login, available space, etc.	1D	Notify Sysadmin	All Virtual Nodes	3.0E+02
IBCORE/IBEDGE	Link up/down, link speed	1D	Notify Sysadmin	Admin Node	2.5E+01
IBEDGE	connectivity to storage	1H	Notify Sysadmin	Admin Node	8.8E-01

Actual Errors Detected
Many Errors are Detected before Catastrophic Application Faults

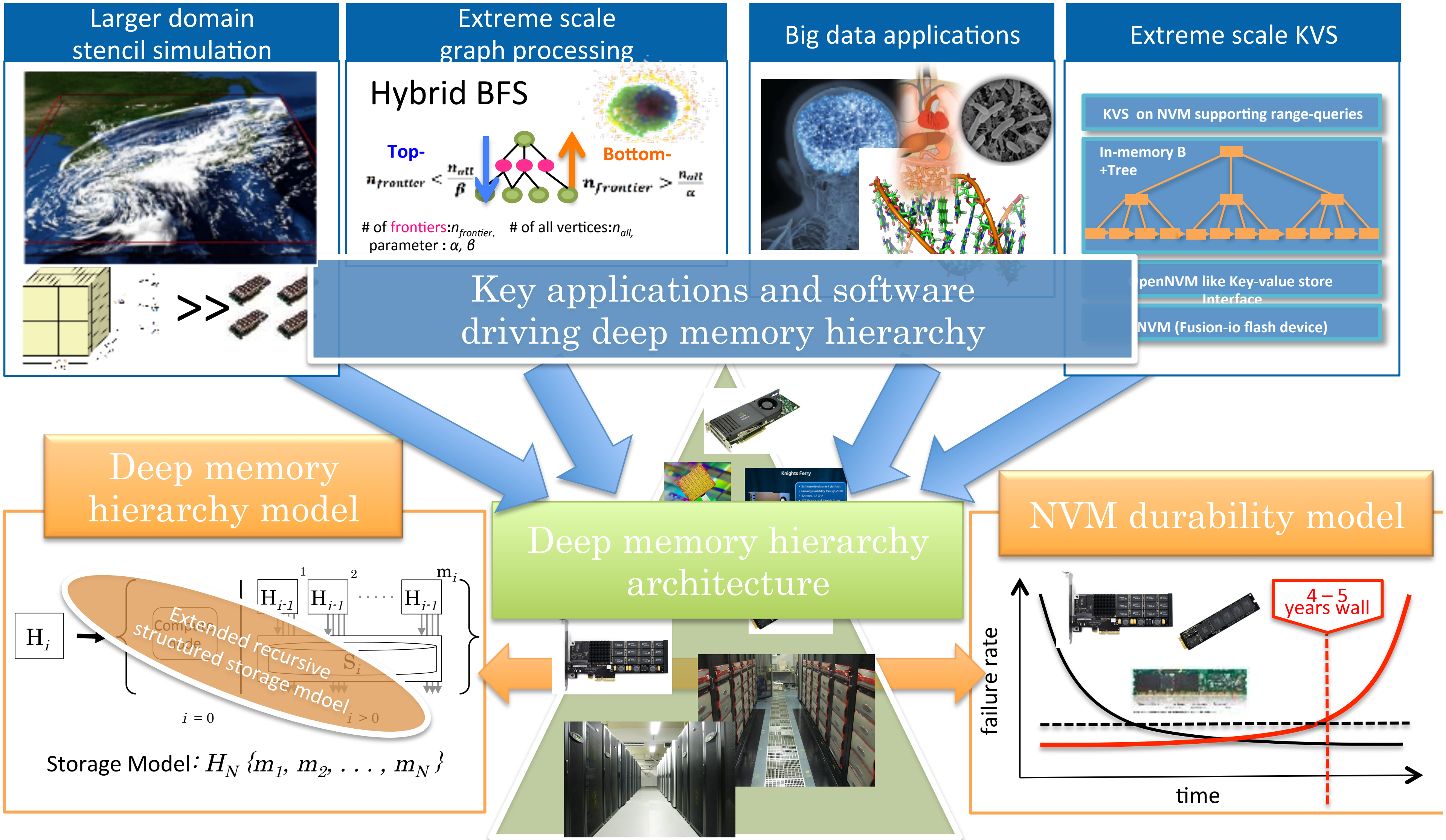
		2012									
		Apr	May	June	July	Aug	Sep	Oct	Nov	Dec	
HDD	Check Available Space	27	90	88	22	14	47	22	44	15	
	Check Mount	49	67	58	81	110	86	12	28	27	
GPU	PCIe Link Speed, Driver Permission, Device Memory ECC Error	31	61	68	46	75	62	32	23	31	
Network	Infiniband Status Check	2	13	18	68	47	25	15	2	4	
SSH	SSH Login	184	217	462	211	256	657	55	26	31	Duplicated Detection
VM Check	SSH Login, Check Available Space & Mount	641	820	638	611	682	2029	753	427	373	Duplicated Detection
Process	Zombie Process	134	5955	481	4378	1692	694	1252	997	9493	Duplicated Detection

		2013					
		Jan	Feb	Mar	Apr	May	
HDD	Check Available Space	21	8	14	0	11	
	Check Mount	29	32	22	68	22	
GPU	PCIe Link Speed, Driver Permission, Device Memory ECC Error	23	46	55	40	31	
Network	Infiniband Status Check	4	7	7	13	3	
SSH	SSH Login	74	27	82	765	35	Duplicated Detection
VM Check	SSH Login, Check Available Space & Mount	517	326	411	145	357	Duplicated Detection
Process	Zombie Process	4305	3505	3320	19408	313	Duplicated Detection

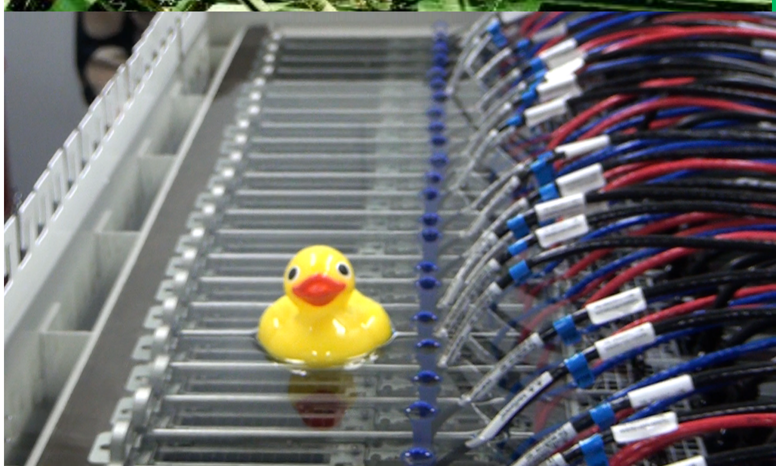
Lessons Learned from Health Checks

- Just like our bodies, a minor system error often does not immediately lead to application failure
- Frequent health checks and corrective actions
  - TSUBAME Storage recovers entirely automatically
- The current HPC failure models nor system software stack does not always have such check & corrective features in a standard way
- We expect TSUBAME3.0 (2Q2016), a ~20 Petaflops machine, to operate in a similar way, scalable to 100+ petaflops range
  - # nodes, components, complexity largely the same

Deep memory hierarchy and modeling



TSUBAME-KFC
(Kepler Fluid Cooling)



A TSUBAME3.0 prototype system
with advanced next gen cooling
40 compute nodes are oil-submerged
1200 liters of oil (Exxon PAO ~1 ton)
#1 Nov. 2013 Green 500!!

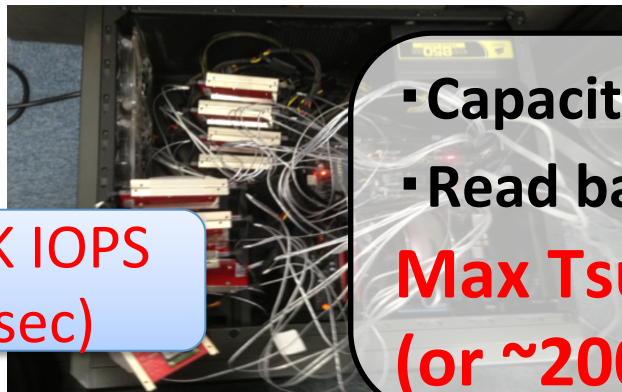
Single Node	5.26 TFLOPS DFP
System (40 nodes)	210.61 TFLOPS DFP 630TFlops SFP
Storage (3SSDs/node)	1.2TBytes SSDs/Node Total 50TBytes ~50GB/s BW

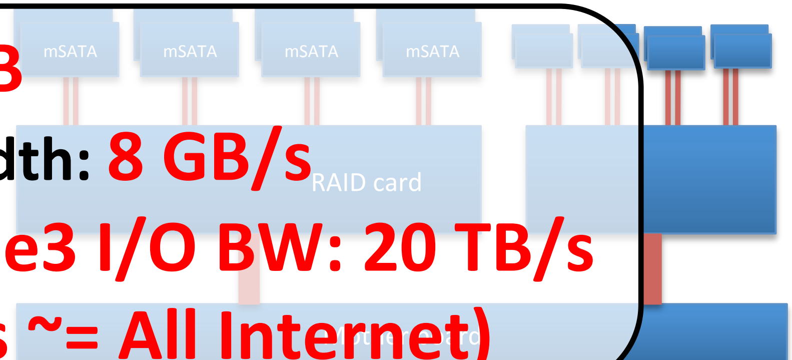
EBD- I/O
(Many-core I/O)

Preliminary I/O Evaluation on GPU and NVRAM

How to design local storage for next-gen supercomputers ?

- Local I/O prototype using 16 mSATA SSDs



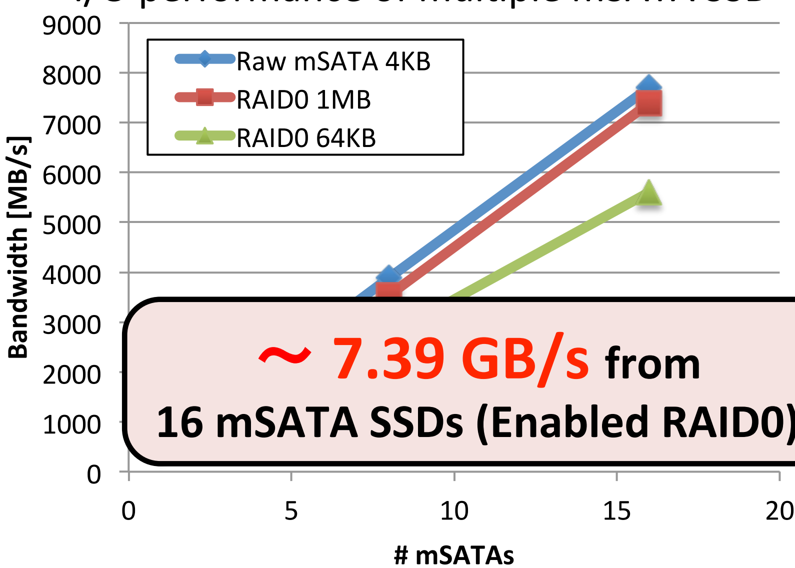


- **Capacity: 4TB**
- **Read bandwidth: 8 GB/s**

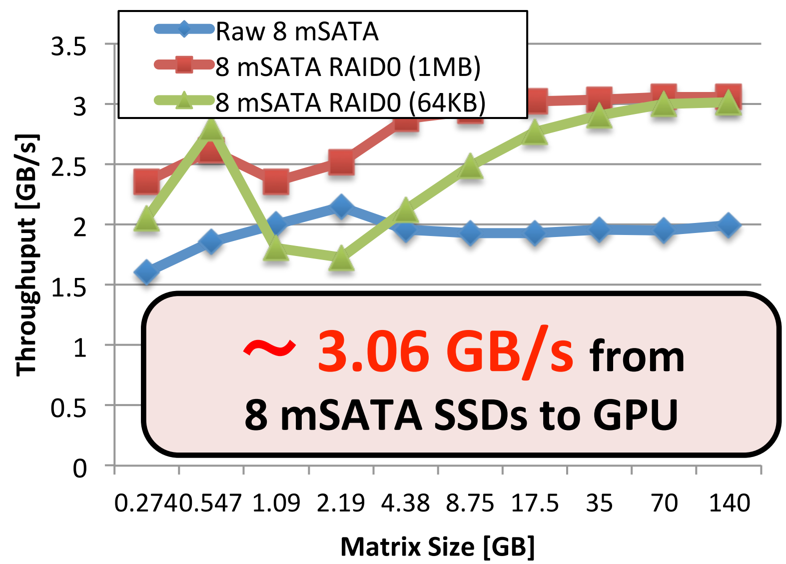
Max Tsubame3 I/O BW: 20 TB/s
(or ~200Tbps ~= All Internet)

～320K IOPS
(3 μ sec)

I/O performance of multiple mSATA SSD

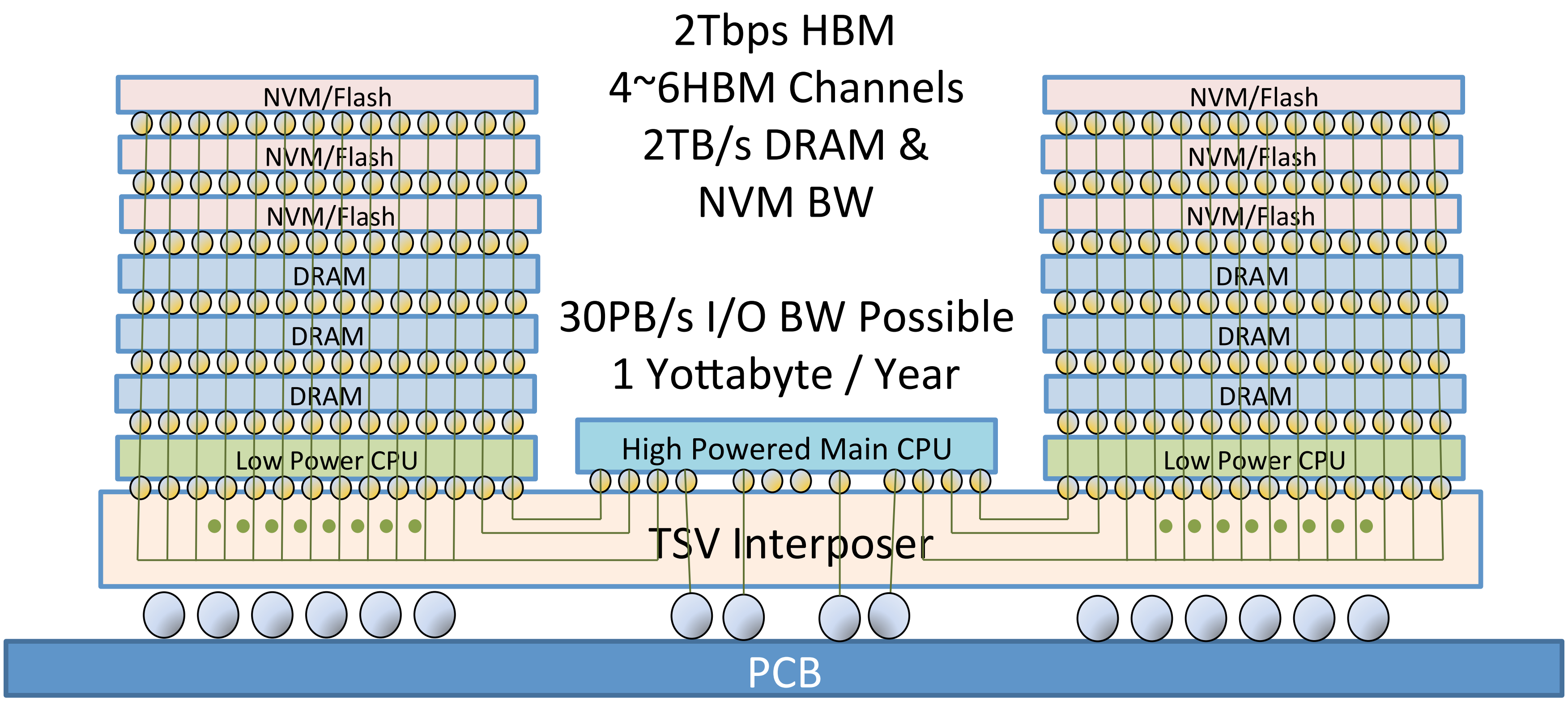


I/O performance from GPU to multiple mSATA SSDs



Tsubame 4: 2021~ DRAM+NVM+CPU with 3D/2.5D Die Stacking

Ultimate Convergence Big Data and Extreme Compute



Direct Chip-Chip Interconnect with planar VCSEL optics

TSUBAME4 2021~ K-in-a-Box (Golden Box) BD/EC Convergent Architecture

1/500 Size, 1/150 Power, 1/500 Cost, x5 DRAM+ NVM Memory

10 Petaflops, 10 Petabyte Hiearchical Memory (K: 1.5PB), 10K nodes

50GB/s Interconnect (200-300Tbps Bisection BW)

(Conceptually similar to HP “The Machine”)

Datacenter in a Box

Large Datacenter will become “Jurassic”

“If it broke don’t fix it” System

- Commoditized HW: aggregation of replace-as-a-whole units
  - **Human repair expensive** => Designing for human repair expensive (c.f. servers vs. smart phones)
  - Redundancy in system design avoiding costly repair for lower aggregate TCO (e.g. RAID)
- **Future SCs and IDCs not subject to post-deployment repairs, but (almost) self-healing**
  - Sufficient redundancy (dark silicon, planar emission photodiodes…) to last the lifetime of a machine (~5 years)
  - Auto-diagnostics with sufficient coverage to automate the process
  - Q: to what extreme can we optimize our system design?
  - Q: what are the SW (+HW) infrastructure necessary?
  - Q: how will Cloud & Big Data apps supported?

GoldenBox Proto1 (NVIDIA K1-based)
To be shown at SC14 Tokyo Tech. Booth…