Argonne NATIONAL LABORATORY

Resilience at Exascale

Marc Snir

Director, Mathematics and Computer Science Division

Argonne National Laboratory

Professor, Dept. of Computer Science, UIUC

U.S. DEPARTMENT OF ENERGY

Problem

- Exascale resilience is “a black swan – the most difficult, under-addressed issue facing HPC.” (ASCAC 2011)
- Fear: a Exaflop/s system will fail so frequently that no useful work will be possible
- DOE & DoD commissioned several reports
  - Inter-Agency Workshop on HPC Resilience at Extreme Scale

    http://institute.lanl.gov/resilience/docs/Inter-AgencyResilienceReport.pdf (Feb 2012)
  - U.S. Department of Energy Fault Management Workshop

    http://shadow.dyndns.info/publications/geist12department.pdf (June 2012)
  - …

Addressing Failures in Exascale Computing

- Week-long workshop summer 2012

M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, A. A. Chien, P. Coteus, N. A. Debardeleben, P. Diniz, C. Engelmann, M. Erez, S. Fazzari, A. Geist, R. Gupta, F. Johnson. S. Krishnamoorthy, S. Leyffer, D. Liberty, S. Mitra, T. Munson, R. Schreiber, J. Stearley, E. V. Hensbergen

- Argonne Report ANL/MCS-TM-332, April 2013.

http://www.mcs.anl.gov/uploads/cels/papers/ANL:MCS-TM-332.pdf

SUPERCOMPUTING TODAY

Argonne Mira (IBM -- Blue Gene/Q)

- 48K nodes
  - 1.6 GHz 16-way core
  - 16 GB RAM
- 768K cores
- 0.768 PB DRAM
- 35 PB Disk storage
  - 240 GB/s bandwidth

- 10 Petaflop/s (10^{16} flop/s) peak performance

- LLNL Sequoia is Mira×2

Oak Ridge Titan

- 18,688 nodes
  - 2.2 GHz AMD 16-core Opteron 6274 processor
  - 32GB DRAM
- 18,688 GPUs
  - NVIDIA Kepler K20
  - 6 GB DRAM
- 299K CPU cores
- 0.71 PB DRAM

- 20 Petaflop/s peak performance

How Reliable Are They?

- MTBF of 1-7 days (failure = lost job)
  - Global system crashes ~1/10 of errors
  - This does not account for failures due to bugs in user code!
- 60%-80% of failures are due to software
  - Mostly in the parallel file system
  - Mostly “performance bugs” (thrashing, time-outs)
- Many complex, cascading errors
  - Root cause analysis is imperfect and very time consuming
- No Byzantine errors
- No silent errors (??)

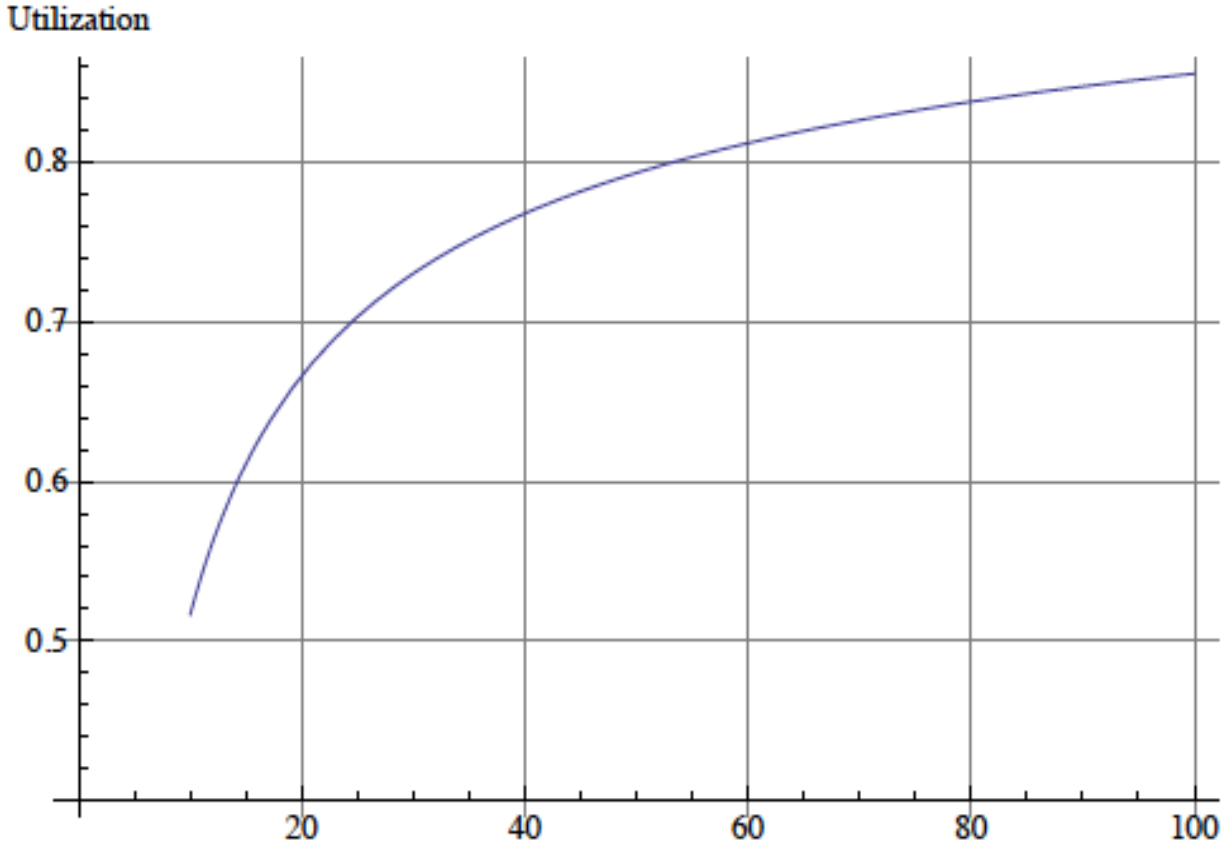
How do we Handle Failures?

- System: Reboot, repair
  - MTTR: 3-24 hours
- Application: Checkpoint, restart
  - User checkpoint/restart
  - ~15-20 minutes checkpoint or restart

- Optimal checkpoint interval

- Chkpt = 15 min, MTBF = 24 hrs => Util ≈ 85%

Utilization, Assuming Poisson Failure Model



Core Assumptions

- Checkpoint time << MTBF (~MTBF/100)
- Recovery time < MTBF (~MTBF/10)
- Errors are detected quickly and are not Byzantine

SUPERCOMPUTING IN 10 YEARS

Exascale Design Point

Systems	2012 BG/Q	2020-2024	Difference Today & 2019
System peak	20 Pflop/s	1 Eflop/s	O(100)
Power	8.6 MW	~20 MW	
System memory	1.6 PB (16*96*1024)	32 - 64 PB	O(10)
Node performance	205 GF/s (16*1.6GHz*8)	1.2 or 15TF/s	O(10) – O(100)
Node memory BW	42.6 GB/s	2 - 4TB/s	O(1000)
Node concurrency	64 Threads	O(1k) or 10k	O(100) – O(1000)
Total Node Interconnect BW	20 GB/s	200-400GB/s	O(10)
System size (nodes)	98,304 (96*1024)	O(100,000) or O(1M)	O(100) – O(1000)
Total concurrency	5.97 M	O(billion)	O(1,000)
MTTI	4 days	O(<1 day)	- O(10)

Both price and power envelopes may be too aggressive!

Going Forward: Risks

- More complex application codes -> more user errors
- More complex system codes -> more “logic” system errors
  - power management, error handling, asynchronous algorithms, dynamic resource provisioning, complex workflows…
- Larger system -> more “performance” system errors
- More hardware -> more hardware errors
- More failure-prone hardware -> More hardware errors
  - Smaller feature size -> more variance, faster aging
  - Sub-threshold logic -> more bit upsets, more multiple-bit upsets

RESILIENCE AT EXASCALE

Core Assumptions

- Checkpoint time << MTBF (~MTBF/100)
- Recovery time < MTBF (~MTBF/10)
- Errors are detected quickly and are not Byzantine

Silent Data Corruption

- Reasonably well studied: Impact of cosmic radiation
- Reasonably easy to protect: DRAM, SRAM, regular arrays of storage)
  - Add more ECC bits and interleave
- Hard to protect: random logic (decoders, ALUs…)
- However:
  - Most (>99%) bit flips have no effect (our HW is inefficient?)
  - Effect is often a hard SW failure

Hardware Error Detection: Assumptions

	45nm	11nm
Cores	8	128
Scattered latches per core	$200,000$	$200,000$
Scattered latchs in uncore relative to cores	$\sqrt{n_{cores}} \times 1.25 = 0.44$	$\sqrt{n_{cores}} \times 1.25 = 0.11$
FIT per latch	10^{-1}	10^{-1}
Arrays per core (MB)	1	1
FIT per SRAM cell	10^{-4}	10^{-4}
Logic FIT / latch FIT	$0.1 - 0.5$	$0.1 - 0.5$
DRAM FIT (per node)	50	50

Hardware Error Detection: Analysis

	Array interleaving and SECDED (Baseline)					
	DCE [FIT]		DUE [FIT]		**UE [FIT]**	
	45nm	11nm	45nm	11nm	45nm	11nm
Arrays	5000	100000	50	20000	1	1000
Scattered latches	200	4000	N/A	N/A	20	400
Combinational logic	20	400	N/A	N/A	0	4
DRAM	50	50	0.5	0.5	0.005	0.005
Total	**1000 - 5000**	**100000**	**10 - 100**	**5000 - 20000**	**10 - 50**	**500 - 5000**
	Array interleaving and ¿SECDED (11nm overhead: $\sim 1\%$ area and $< 5\%$ power)					
	DCE [FIT]		DUE [FIT]		**UE [FIT]**	
	45nm	11nm	45nm	11nm	45nm	11nm
Arrays	5000	100000	50	1000	1	5
Scattered latches	200	4000	N/A	N/A	20	400
Combinational logic	20	400	N/A	N/A	0.2	5
DRAM	50	50	0.5	0.5	0.005	0.005
Total	**1500 - 6500**	**100000**	**10 - 50**	**500 - 5000**	**10 - 50**	**100 - 500**
	Array interleaving and ¿SECDED + latch parity (45nm overhead $\sim 10\%$; 11nm overhead: $\sim 20\%$ area and $\sim 25\%$ power)					
	DCE [FIT]		DUE [FIT]		**UE [FIT]**	
	45nm	11nm	45nm	11nm	45nm	11nm
Arrays	5000	100000	50	1000	1	5
Scattered latches	200	4000	20	400	0.01	0.5
Combinational logic	20	400	N/A	N/A	0.2	5
DRAM	0	0	0.1	0.0	0.100	0.001
Total	**1500 - 6500**	**100000**	**25 - 100**	**2000 - 10000**	**1**	**5 - 20**

Summary of (Rough) Analysis

- If no new technology is deployed can have up to one undetected error per hour
- With additional circuitry could get down to one undetected error per 100-1,000 hours (week – months)
  - Similar to what we have now!
- With no new invention, cost is about 20% additional circuits and 25% additional power
  - New invention may reduce overhead
- *Not clear required components will be available at low cost*
  - Market for highly reliable servers is not growing
  - Fastest growing markets (mobile, consumer products, clouds) requires low power & low cost but do not require high availability

SW Alternatives to HW Error Detection

- Replicate execution (for critical, rarely executed code – e.g., system code)
  - Can cost << x2, with architecture/compiler support (assuming memory is trusted)
- Add (via compilation) program level property checks
  - SWAT project (S. Adve): 85% coverage of SDCs with 10% overhead
- Add error detection to application code (e.g., redundancy in dense linear algebra)
- Develop fault-tolerant algorithms
- *Hypothesis*: bit flips
  - Either destroy the compute model abstraction (wrong pointers, wrong jump addresses) – and can very often be detected
  - Or can be treated as noise in the computation – and handled algorithmically

Core Assumptions

- Checkpoint time << MTBF (~MTBF/100)
- Recovery time < MTBF (~MTBF/10) [<< 1 hour]
- Errors are detected quickly and are not Byzantine

Recovery Time

- Localized failure (e.g., node failure)
  - Replace node and restart application from checkpoint – seconds – minutes
- Global system crash
  - Switch, parallel file system, resource manager, monitoring & control SW…
  - Often combination of HW failure and SW "performance bug"
  - May take hours to recover
- *Need global OS services that are more resilient or recover much faster (OS/R proposal)*
  - APIs for resilience (reliable execution, reliable storage)
  - Hierarchical error handling (fault containment)
  - Reliable pub-sub service for reliability-related events

Core Assumptions

- Checkpoint time << MTBF (~MTBF/100) [< 1 min]
- Recovery time < MTBF (~MTBF/10)
- Errors are detected quickly and are not Byzantine

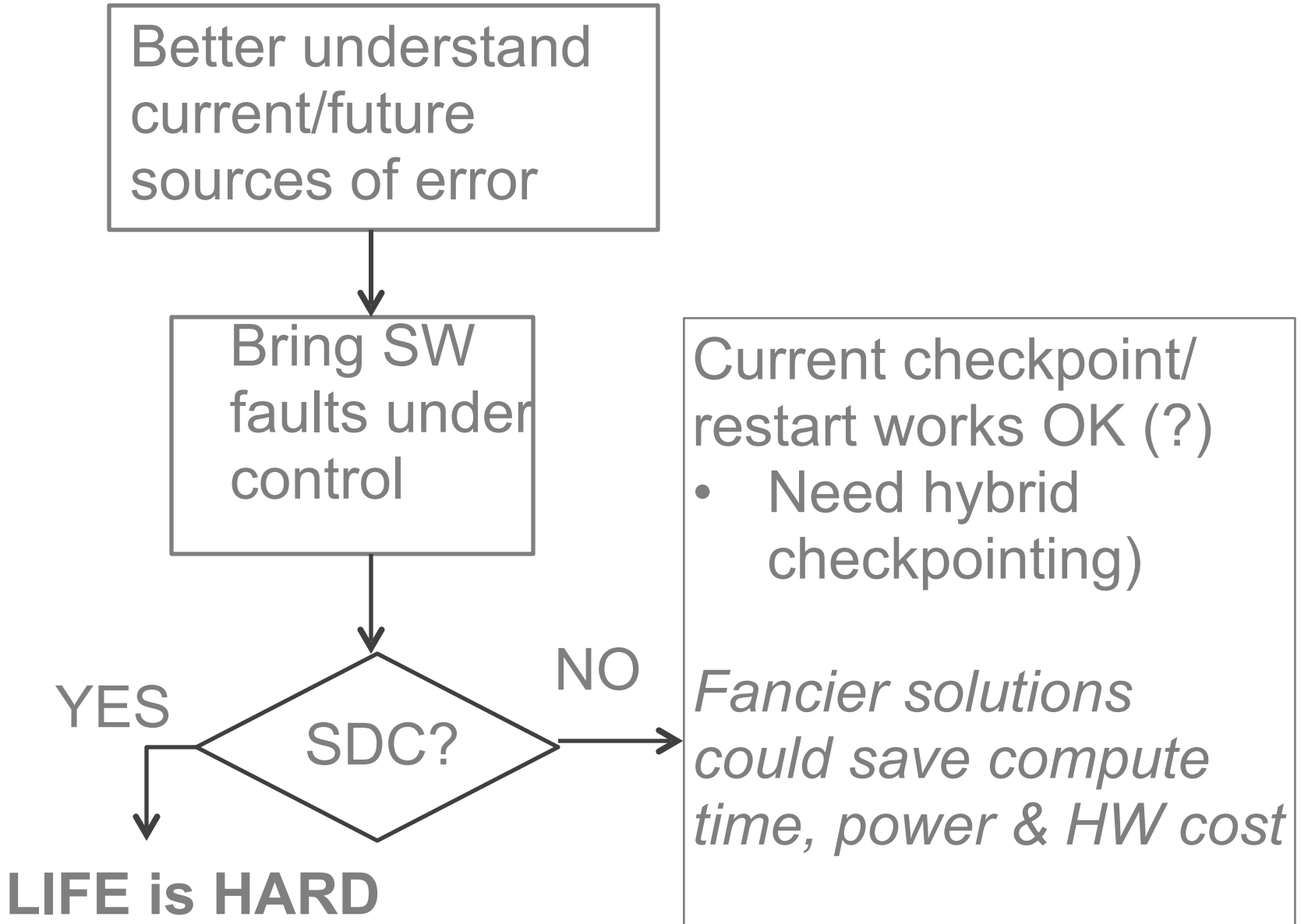
Hybrid Checkpoint

- Fast, frequent checkpoints to take care of frequent failures; slower, less frequent checkpoint to take care of less frequent failures
- Checkpoint in memory: handles transient errors
  - Seconds; need more memory (~50%) but no significant additional power
- Checkpoint in NVRAM memory: can handle node failure, if "twin tailed"
- Checkpoint in memory + RAID5 – handle node failures
  - ~3 minutes; ~50% more memory
- Checkpoint in remote NVRAM ("burst buffers")
- Checkpoint on disk
- Doable – but may be expensive and may be hard if node memory is much larger (LBNL, ANL)

Avoid Global Checkpoints

- Cluster checkpoint + logging
  - Can avoid domino effect of uncoordinated checkpoints for send-deterministic apps
- Save energy and recovery time
- Containment domains
- Algorithmic error correction
  - Use redundancy in computation state
- …

- *Are these techniques general?*


Better understand current/future sources of error
Bring SW faults under control
SDC?
YES
NO
LIFE is HARD
Current checkpoint/ restart works OK (?)
• Need hybrid checkpointing)
Fancier solutions could save compute time, power & HW cost

Life with SDCs

- Build system SW immune to SDCs or build build good detectors and fast repair
- Build middleware (compilers, run-time) that can detect and correct “abstraction breaking” SDCs in user code
- Built application SW that detects SDCs in data or can tolerate them
- Build infrastructure to compose everything


The End
?