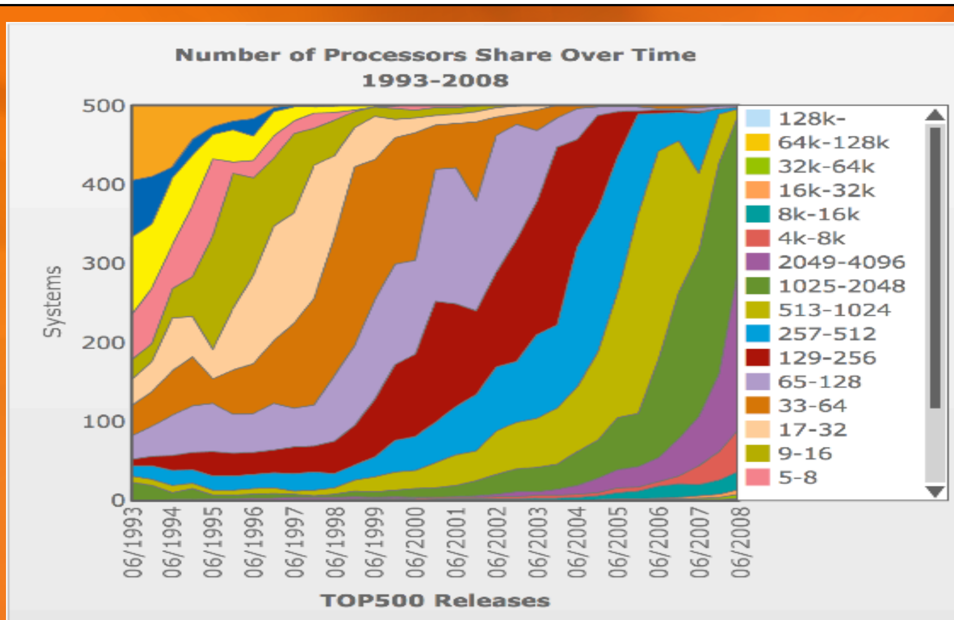
Surviving in the Petascale World [and Beyond]

George Bosilca
Innovative Computing Laboratory
University of Tennessee

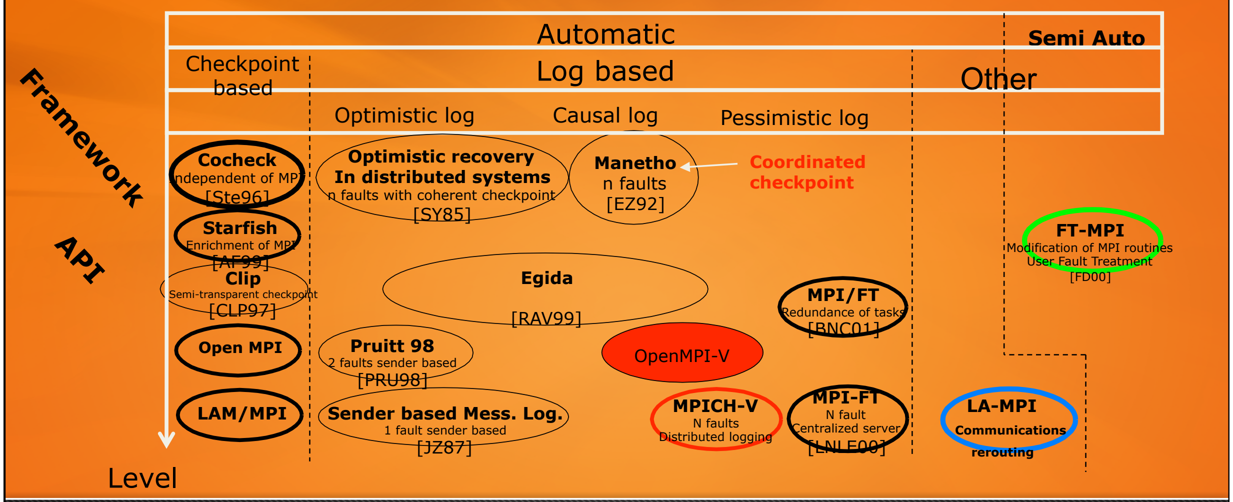
ICL UT
INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science



ICL UT

Classification of FT approaches



FT a complex solution

Transparency

- application ckpt: application stores intermediate results and restart form them
- MP API+FM: message passing API returns errors to be handled by the programmer
- automatic: runtime detects faults and handle recovery

Checkpoint coordination

- coordinated: all processes are synchronized, network is flushed before ckpt; all processes rollback from the same snapshot
- uncoordinated: each process checkpoint independently of the others; each process is restarted independently of the others

FT a complex solution

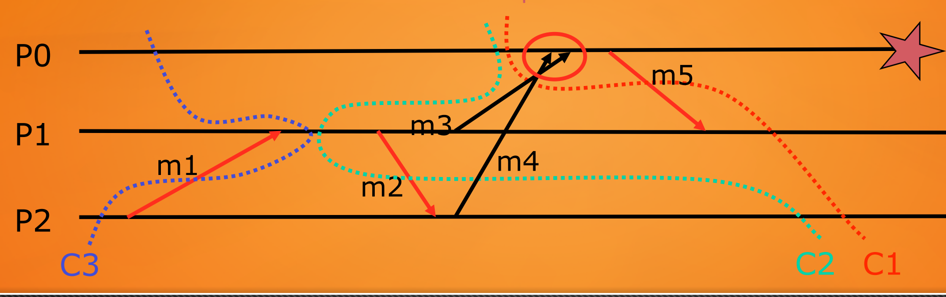
Message logging

- pessimistic: all messages are logged on reliable media and used for replay
- optimistic: all messages are logged on non reliable media. If 1 node fails, replay is done according to other nodes logs. If >1 node fail, rollback to last coherent checkpoint
- causal: optimistic + Antecedence Graph, reduces the recovery time

ICL UT

The problem of inconsistent states

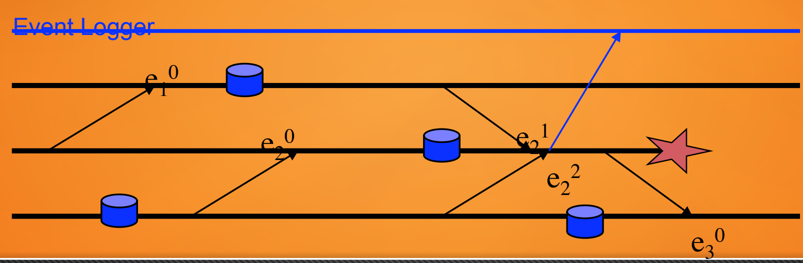
- Order of message receptions are non-deterministic events
- messages received but not sent are inconsistent
- Domino effect can lead to rollback to the begining of the execution in case of even a single fault
- Possible loose of the whole execution and unpredictive fault cost



ICL UT

Deterministic Recovery

- Deterministic replay is based on ***Event Logging***
- ***Piecewise Deterministic*** assumption (even suitable for monte carlo applications)
- Each recv is an event (src,send-clk,recv-clk)
- Send the ordering of events to stable storage (event logger)

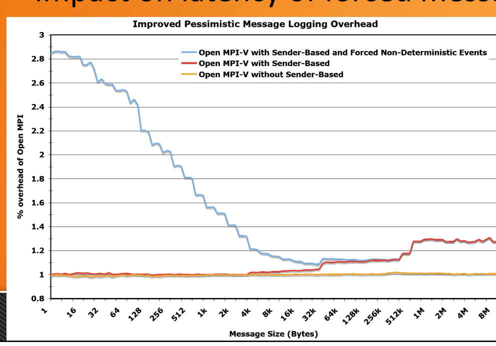


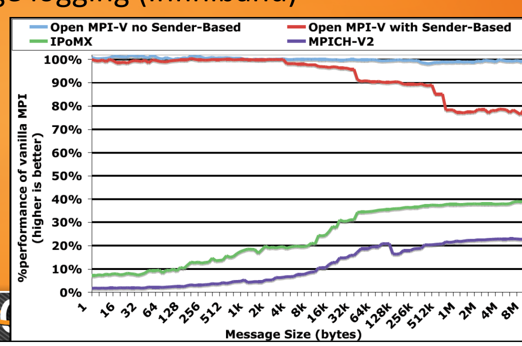
Benchmark Performance

- Number of logged events to total number of messages

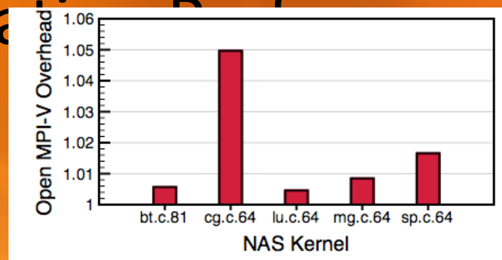
	BT	SP	FT	CG	MG						LU					
#processors	all				4	32	64	256	512	1024	4	32	64	256	512	1024
%non-deterministic	0	0	0	0	40.33	29.35	27.10	22.23	20.67	19.99	1.13	0.66	0.80	0.80	0.75	0.57

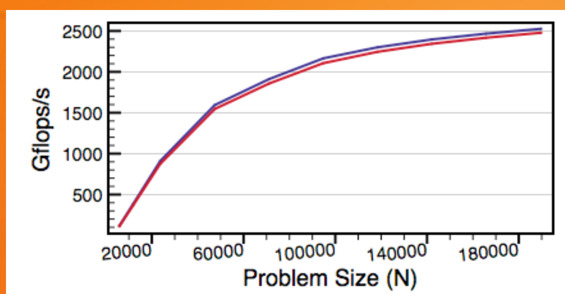
- Impact on latency of forced message logging (Infiniband)

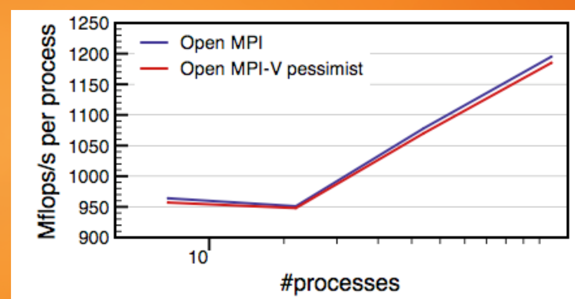




Applica… ce and







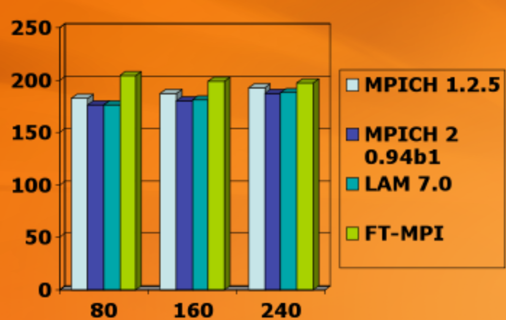
FT-MPI: Why and How ?

- MPI is the de-facto programming model for parallel applications
- MPI Standard: *"Advice to implementors: A good quality implementation will, to the greatest possible extent, circumvent the impact of an error, so that normal processing can continue after an error handler was invoked."*
- Define the behavior of MPI [state] in case an error occurs and give the application the possibility to recover from a node-failure
- A regular, non fault-tolerant MPI program will run using FT-MPI
- Follows the MPI-1 and MPI-2 specification as closely as possible (e.g. no additional function calls)
- On error user program must do something (!)

Recovery modes

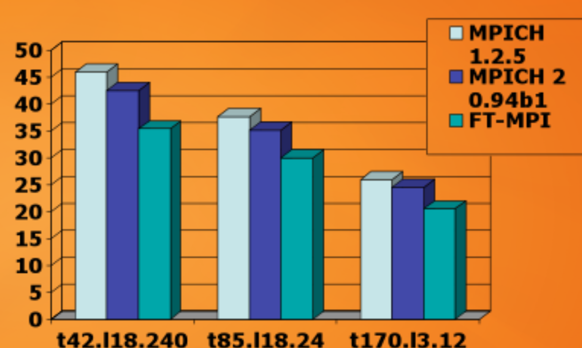
- ABORT, BLANK, SHRINK and REBUILD
- REBUILD: a new process is created, and it will return MPI_INIT_RESTARTED_PROC from MPI_Init
- BLANK: dead processes replaced by MPI_PROC_NULL, all communications with such a process succeed, they do not participate in the collectives
  - two sub-modes: local and global

Shallow Water (PSTSWM) & HPL

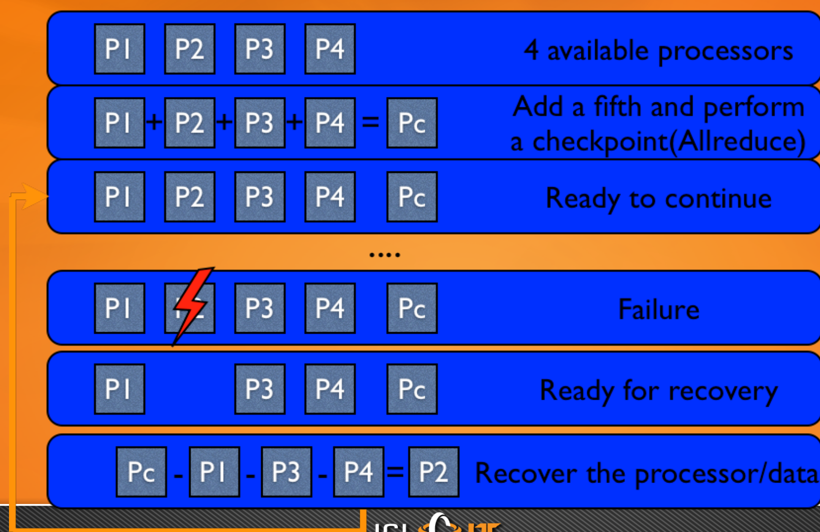


HPL

PSTSWM



Diskless checkpointing



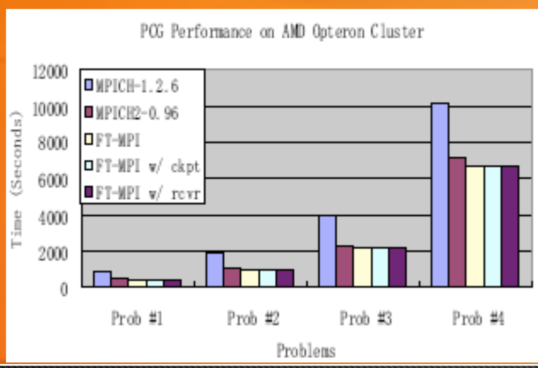
Diskless Checkpointing

- How to checkpoint ?
  - either floating-point arithmetic or binary arithmetic will work
  - If checkpoints are performed in floating-point arithmetic then we can exploit the linearity of the mathematical relations on the object to maintain the checksums
- How to support multiple failures ?
  - Reed-Salomon algorithm
  - support p failures require p additional processors (resources)

PCG

Fault Tolerant CG

	Size of the Problem	Num. of Comp. Procs
Prob #1	164,610	15
Prob #2	329,220	30
Prob #3	658,440	60
Prob #4	1,316,880	120

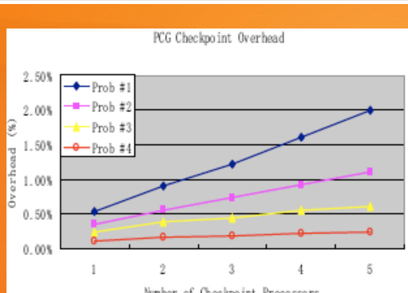


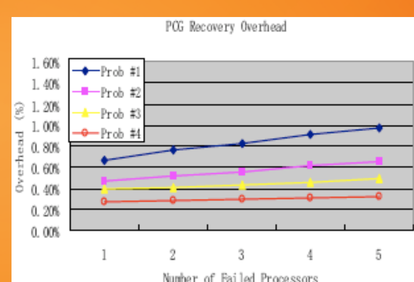
- Performance of PCG with different MPI libraries

PCG

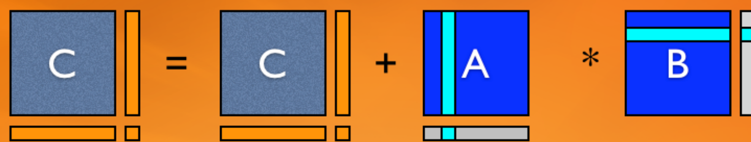
Time	Prob #1	Prob #2	Prob #3	Prob #4
1 ckpt	2.6	3.8	5.5	7.8
2 ckpt	4.4	5.8	8.5	10.6
3 ckpt	6.0	7.9	10.2	12.8
4 ckpt	7.9	9.9	12.6	15.0
5 ckpt	9.8	11.9	14.1	16.8

Checkpoint overhead in seconds





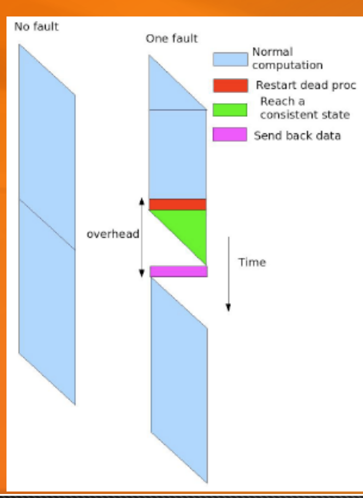
ABFT-PDGEMM



PDGEMM-SUMMA	ABFT-PDGEMM-SUMMA
$\frac{2n^3}{P}\gamma + 2(n + 2\sqrt{P} - 3)(\frac{n}{\sqrt{P}}\beta)$	$\frac{2n(n+nloc)^2}{P}\gamma + 2(n + 2\sqrt{P} - 3)(\frac{(n+nloc)}{\sqrt{P}}\beta)$

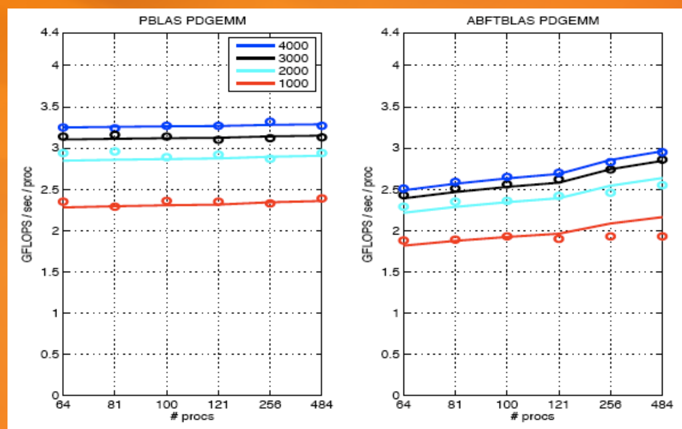
- The algorithm maintain the consistency of the checkpoints of the matrix C naturally

Failure Overhead

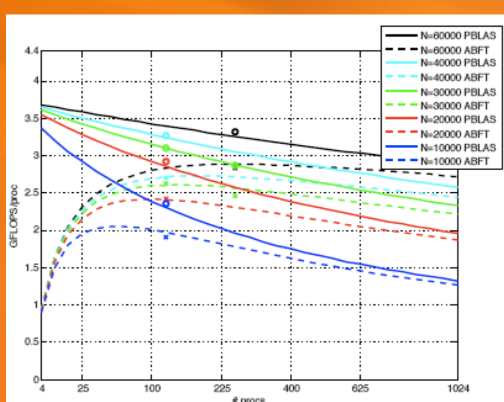


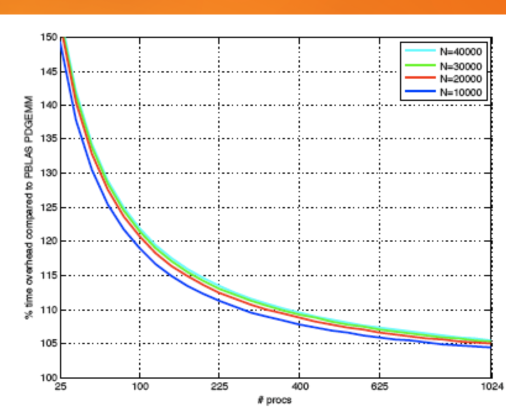
- FT-MPI will take care of the fault management
- Once the new process joins the MPI_COMM_WORLD we have to rebuild the communicators
- Then we have to retrieve the data from the checkpoint processor

PBLAS vs. ABFT BLAS (no failure)



Strong Scalability





Conclusion

- Fault Tolerance is a requirement
- Which model is the best depend on many factors
  - FT-MPI is a viable approach with algorithms already available.