2014

« Advanced Multi Fault Tolerance » in parallel applications : Multi level storage with FTI middleware

Eric Boyer - CINES

Involved Partners

With the support of :

Context (1/3)

Fault tolerance and application resiliency will be a key issue for next multi-petascale and Exascale system as identified by reports including IESP and EESI.

Technological evolution makes this challenge difficult to achieve

Context (2/3)

Checkpointing at system level on multi-petascale and on coming exascale systems is no longer suitable :

100’s GB memory in 2004 top10 HPC

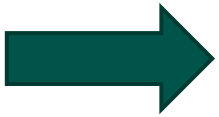
PB of memory in 2014 …

Tiered storage involved (local and global) will not handle such amount and related throughput. Application level, targeting pertinent information is the appropriate option.

Context (3/3)

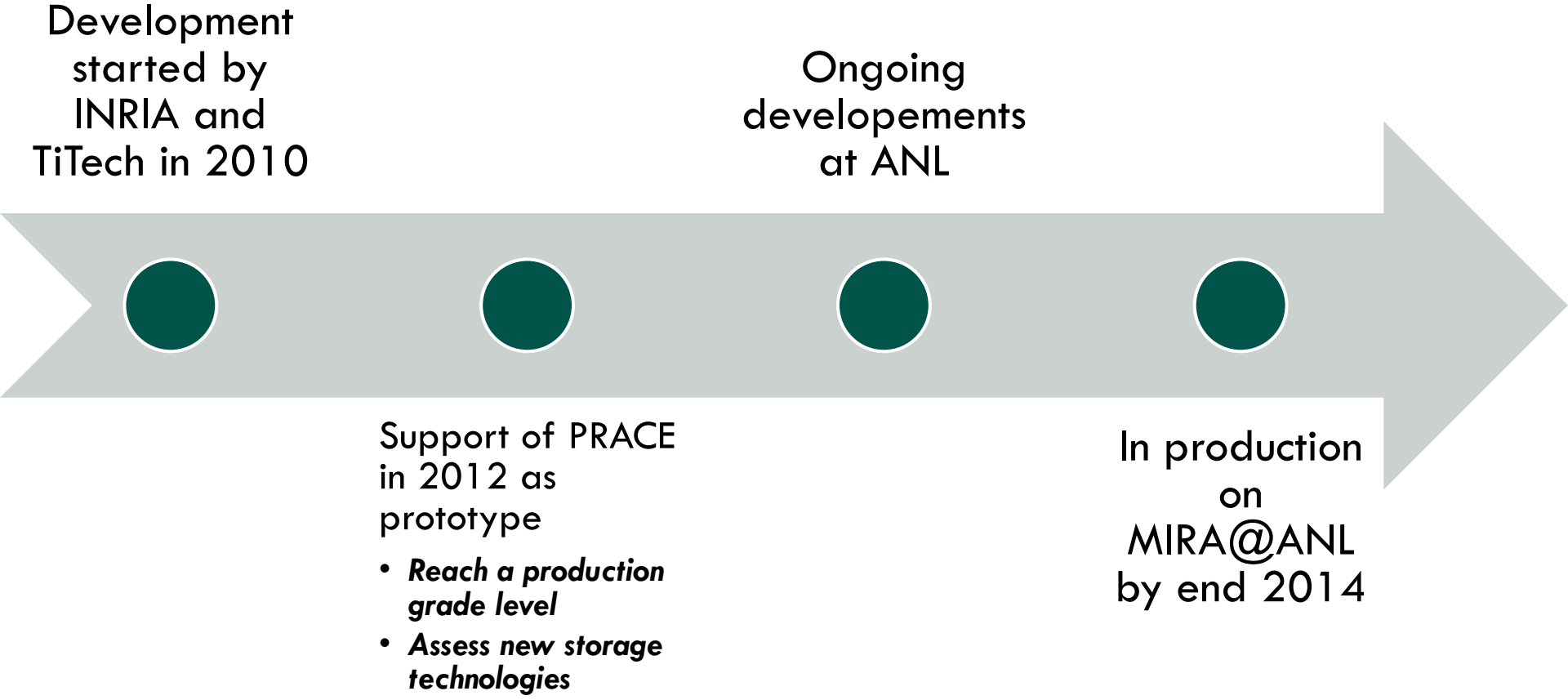
Even at application level, major pitfalls remain !

Increase of hardware component lead to low MTTI, involving increased checkpoint frequency



On exascale systems, this should lead to an unaffordable contribution of checkpoint creation on application execution time.

FTI : Fault Tolerance Interface

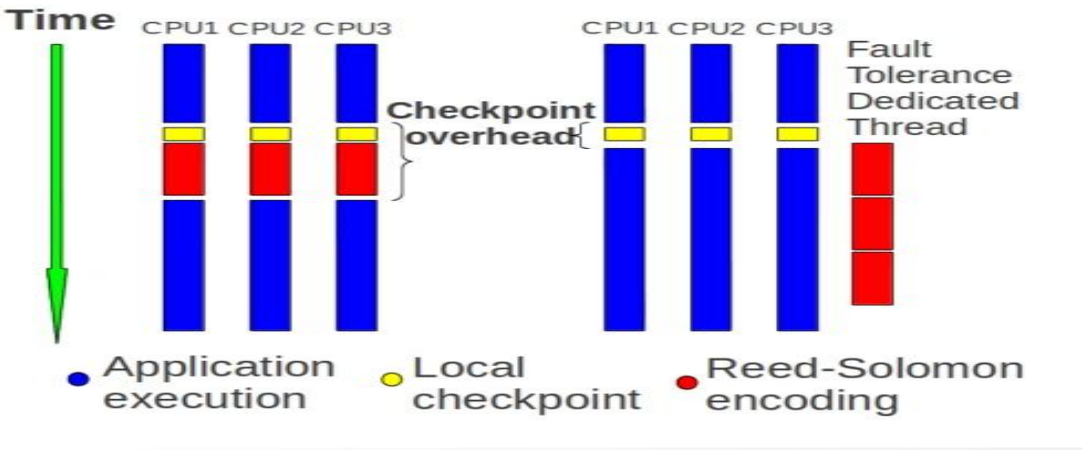


FTI : Fault Tolerance Interface

- Suitable at application level
  - Easy to implement on already checkpointed applications
  - The application integration focuses on « what »
  - FTI handles « how » and takes the best advantage of the HPC platform
  - Embedded added value by FTI hardly integrable by application implementation

FTI : Fault Tolerance Interface

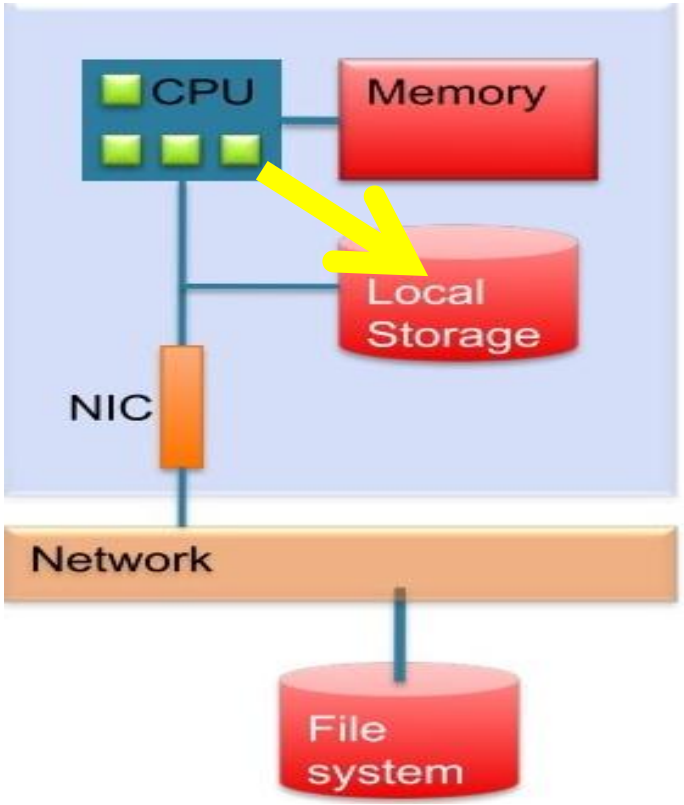
- Introduces asynchronous processus for checkpoint



- Relies on multi-level storage technologies

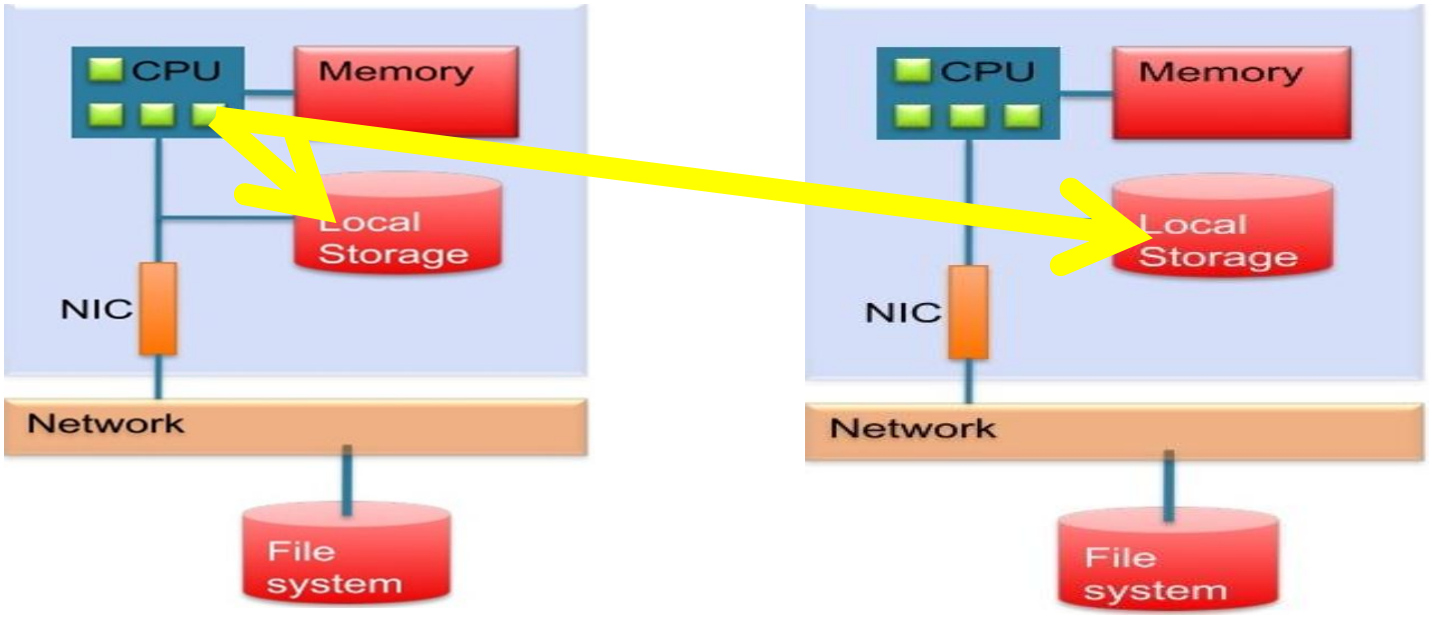
FTI : Protection levels

- **L1** : FTI performs checkpointing on local storage without any concern to resist a failure of this support.



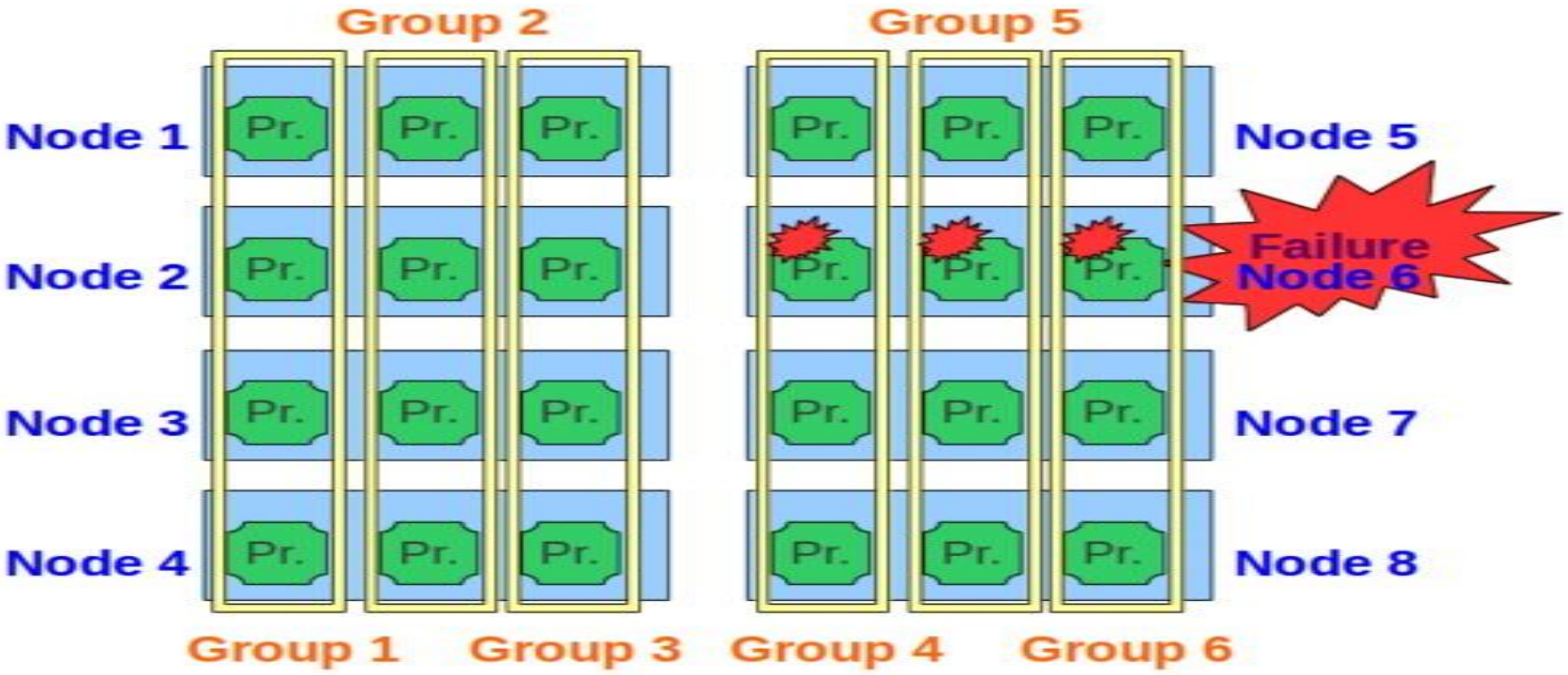
FTI : Protection levels

- **L2** : FTI does a local checkpoint and duplicates to a partner node. This mode allows an application to resume from any failure involving one node in each partnership.



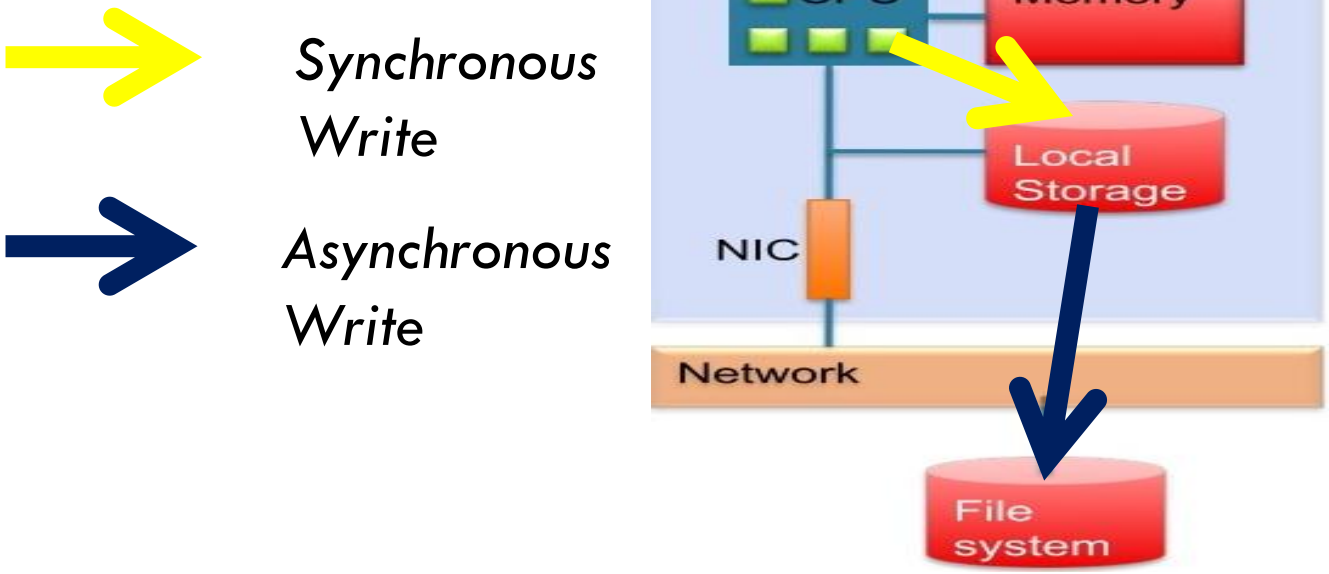
FTI : Protection levels

- **L3** :FTI dispatches checkpoints to a group of multiple nodes. The backup is encoded with other processes using a Reed-Solomon algorithm.

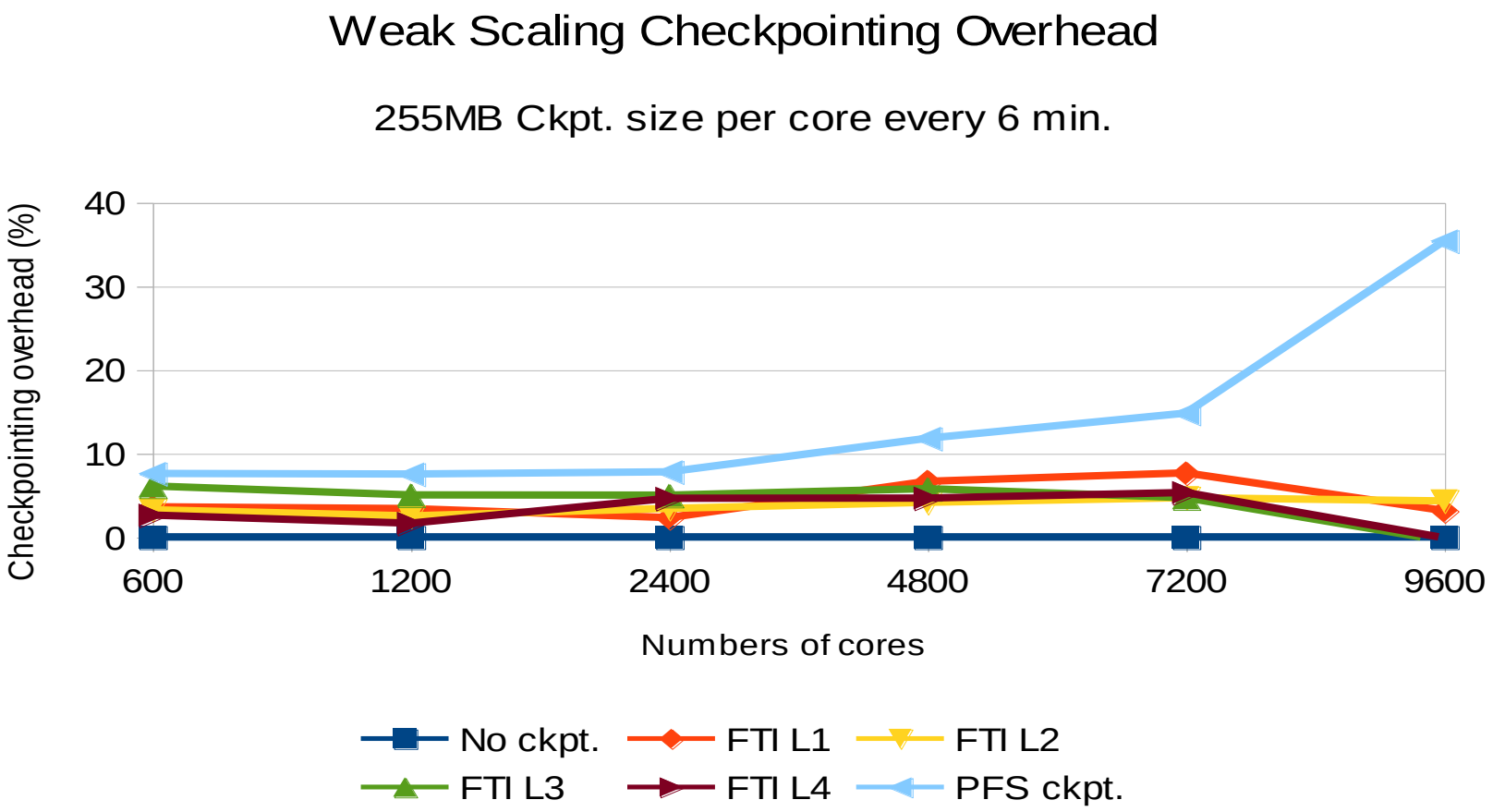


FTI : Protection levels

- **L4** : FTI writes the checkpoint on the parallel file system (PFS). To hide the cost of the backup level, the local storage system is used as a buffer to write the data on PFS asynchronously.

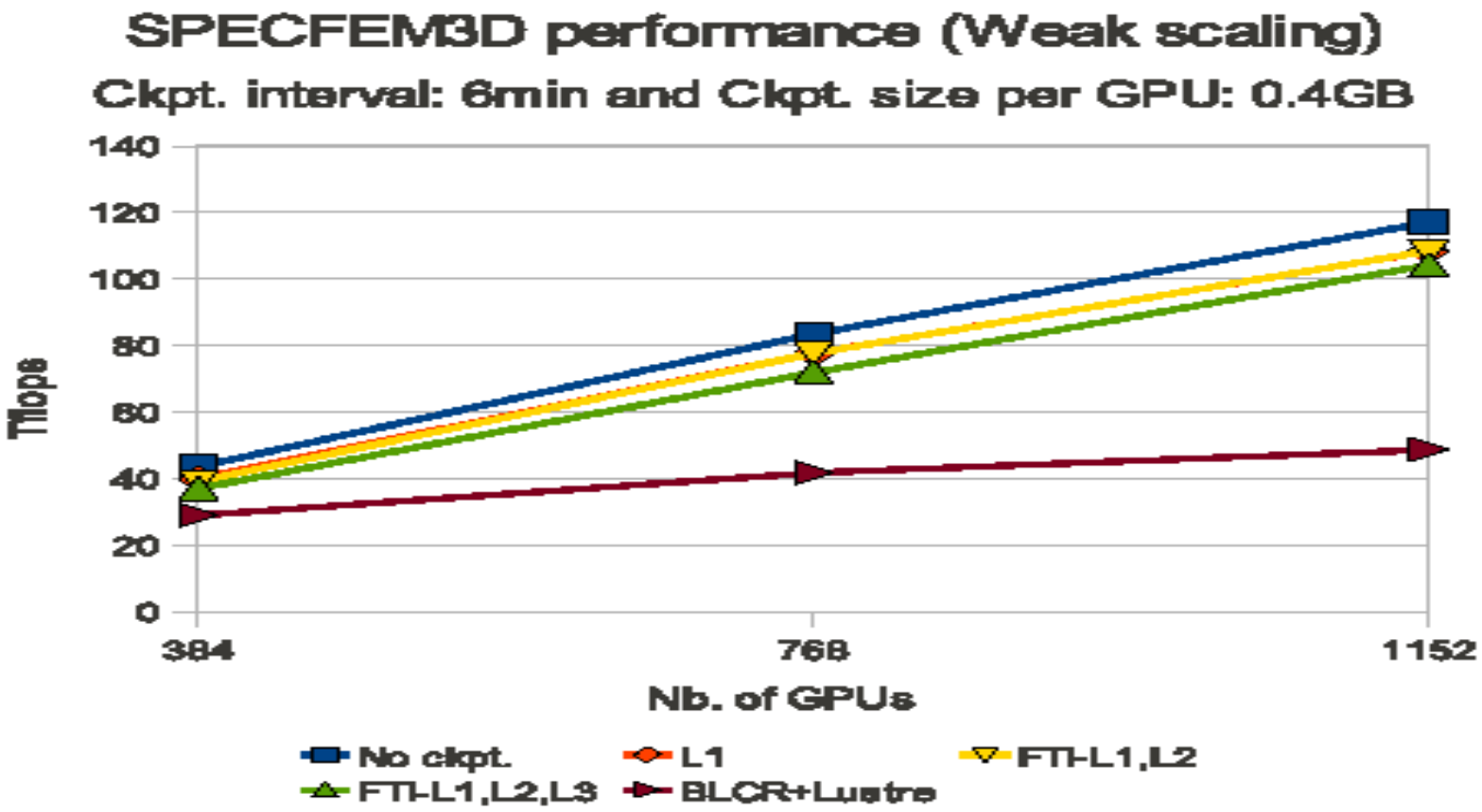


FTI : HYDRO



CURIE – TGCC / Performed by CINES

FTI : SPECFEM3D

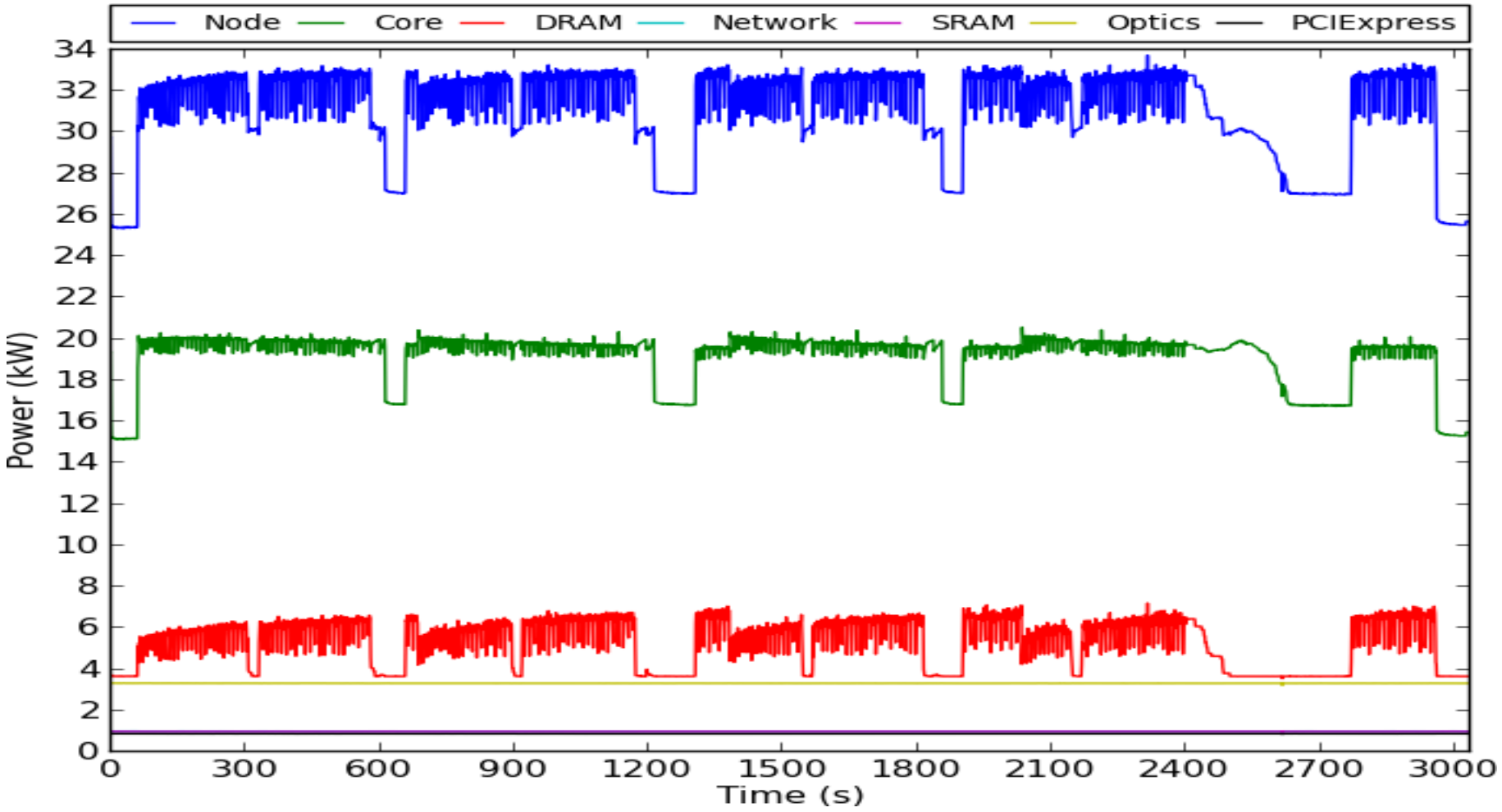


TSUBAME 2.0 / TiTech

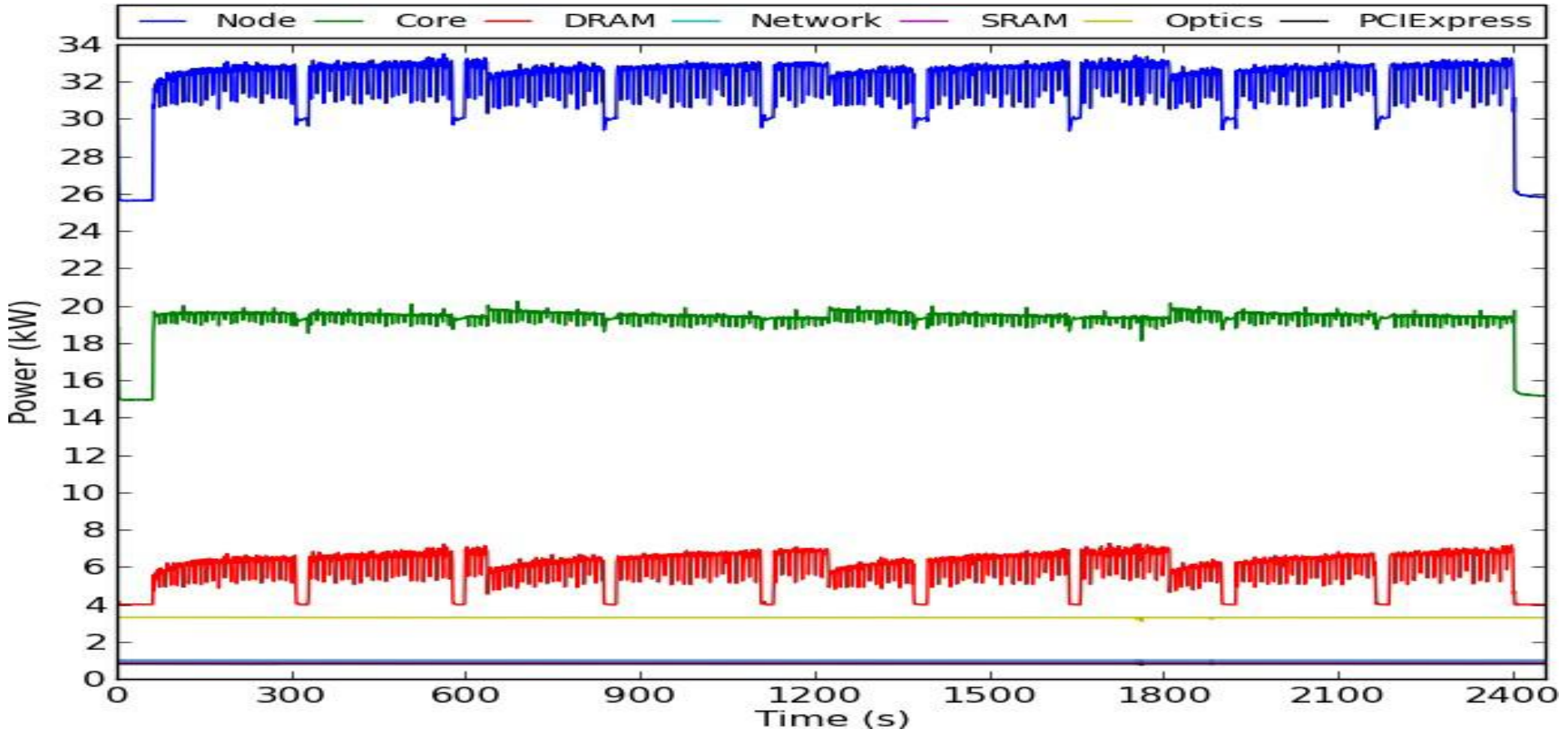
FTI : LAMMPS

- Integrated LAMMPS with FTI
- Lennard-Jones simulation
- 1.3 billion particles
- 32,678 MPI ranks on MIRA (BG/Q)
- Ckpt. interval : L1:4, L2:8, L3:16, L4:32
- Ckpt. pattern : 1-2-1-3-1-2-1-4

FTI : LAMMPS – Synchronous mode



FTI : LAMMPS – Asynchronous mode



Speedup 1,2 / Synchronous mode

FTI : GYSELA5D

- Contribution of « Maison de la Simulation »
- Work in progress …
- Test bed for enhancements and future developments :

➔ Towards a production grade solution (DONE)

- Introduction of a Fortran API
- Removal of the dependency on python
- Clean-up of the API (no more use of global variables, ...)
- Support for a forced checkpoint at last iteration for multi-job runs
- Support of a degraded mode with no dedicated FTI process
- FTI errors can not stop the application anymore

➔ Performance evaluation in a production code (WIP)

- Evaluation on multiple real-life crash scenarii
- Comparison with the use of local memory rather than local storage
- Analysis of the impact of the dedicated process without dedicated core

FTI : Implementation

FTI is a collection of few simple functions to call within the application :

•int FTI_Init (char _configFile, MPI_Comm globalComm);
this function will initialize FTI.

•int FTI_Protect (int id, void _ptr, long size);
It stores a pointer to a variable that needs to be protected.

•int FTI_Snapshot ();
*this function t*akes an FTI snapshot or recover the data if it is a restart.

•int FTI_Finalize ();
this function closes FTI properly on the application processes.

FTI : Implementation

- **FTI_Init:**
  - Read configuration file
  - Creates checkpoint directories
  - Detect topology of the system
  - Regenerate data upon recovery

FTI : Implementation

- **FTI_Protect:**
  - Stores metadata concerning the variable to protect

FTI : Implementation

- **FTI_Snapshot:**
  - Test if it is time for a checkpoint
  - If it is, it checks which level of ckpt.
  - It saves the checkpoint as requested
  - It loads the checkpoint upon recovery

FTI : Implementation

- **FTI_Finalize:**
  - Frees the allocated memory
  - Informs it is over to dedicated threads
  - Clean checkpoints and metadata

FTI : How to implement ?

At application level :

- Selection of data to protect to initiate a restart in case of failure

At submission time :

- Set up in « config.fti » file
  - Storage locations for checkpointed data
  - Levels / combination
  - Async / sync
  - Frequency or period

FTI : Next steps ?

- Silent data corruption handling

Behavioral-based SDC detection

- Spacial analysis
- Temporal analysis
- Approximate computing

- Checkpoint data compression

Credits

- Leonardo Bautista Gomez (ANL)
- Julien Bigot (Maison de la Simulation)
- Franck Capello (ANL)
- Guillaume Colin de Verdière (CEA)
- Stéphane Requena (GENCI)
- Adèle Villermet (Student@ENSEIRB-MATMECA)

REFERENCE

FTI is Open Source under LGPL

http://leobago.com/projects/fti

leobago@mcs.anl.gov

QUESTIONS ?

Backup Slides

- IESP¨: International Exascale Software Project : http://www.exascale.org/iesp/IESP
- EESI : European Exascale Sotware Initiative : http://www.eesi-project.eu
- MTTI : Mean Time To Interrupt

Backup Slide : Tera to Exa / systems

System attributes	2010	"2015"		"2018"		Difference Today & 2018
System peak	2 Pflop/s	200 Pflop/s		1 Eflop/sec		O(1000)
Power	6 MW	15 MW		~20 MW		
System memory	0.3 PB	5 PB		32-64 PB		O(100)
Node performance	125 GF	0.5 TF	7 TF	1 TF	10 TF	O(10) – O(100)
Node memory BW	25 GB/s	0.1 TB/sec	1 TB/sec	0.4 TB/sec	4 TB/sec	O(100)
Node concurrency	12	O(100)	O(1,000)	O(1,000)	O(10,000)	O(100) – O(1000)
Total Concurrency	225,000	$O(10^8)$		$O(10^9)$		O(10,000)
Total Node Interconnect BW	1.5 GB/s	20 GB/sec		200 GB/sec		O(100)
MTTI	days	O(1day)		O(1 day)		- O(10)

Backup slide : Tera to Exa /software

- **Steepness of the ascent from terascale to petascale to exascale**
- Extreme parallelism and hybrid design
  - Preparing for million/billion way parallelism
- Tightening memory/bandwidth bottleneck
  - Limits on power/clock speed implication on multicore
  - Reducing communication will become much more intense
  - Memory per core changes, byte-to-flop ratio will change
- Necessary Fault Tolerance
  - MTTF will drop
  - Checkpoint/restart has limitations

Backup Slide : FTI screen shot in HYDRO

```
  //pre-allocate memory before entering in loop
  //For godunov scheme
  start = cclock();
  start = cclock();
  allocate_work_space(H.nxyt, H, &Hw_godunov, &Hvw_godunov);
  compute_deltat_init_mem(H, &Hw_deltat, &Hvw_deltat);
  end = cclock();
#ifdef MPI
#if FTI==1
  FTI_Protect(0,functim, TIM_END,FTI_DBLE);
  FTI_Protect(1,&nvtk,1,FTI_INTG);
  FTI_Protect(2,&next_output_time,1,FTI_DBLE);
  FTI_Protect(3,&dt,1,FTI_DBLE);
  FTI_Protect(4,&MflopsSUM,1,FTI_DBLE);
  FTI_Protect(5,&nbFLOPS,1,FTI_LONG);
  FTI_Protect(6,&(H.nstep),1,FTI_INTG);
  FTI_Protect(7,&(H.t),1,FTI_DBLE);
  FTI_Protect(8,Hv.uold,H.nvar * H.nxt * H.nyt,FTI_DBLE);
#endif
#endif
  if (H.mype == 0) fprintf(stdout, "Hydro: init mem %lfs\n", ccelaps(start, end));
  // we start timings here to avoid the cost of initial memory allocation
  start_time = dcclock();

  while ((H.t < H.tend) && (H.nstep < H.nstepmax)) {
    // reset perf counter for this iteration
    flopsAri = flopsSqr = flopsMin = flopsTra = 0;
    start_iter = dcclock();
    outnum[0] = 0;
    if ((H.nstep % 2) == 0) {
      dt = 0;
```

Backup Slide : FTI screen shot in HYDRO

```
    if (H.mype == 0) {
      fprintf(stdout, "--> step=%4d, %12.5e, %10.5e %s\n", H.nstep, H.t, dt, outnum);
      fflush(stdout);
    }
#ifdef MPI
#if FTI==1
    // FTI snapshot
    FTI_Snapshot();
#endif
#endif
  } // while
  end_time = dcclock();

  // Deallocate work spaces
  deallocate_work_space(H.nxyt, H, &Hw_godunov, &Hvw_godunov);
  compute_deltat_clean_mem(H, &Hw_deltat, &Hvw_deltat);

  hydro_finish(H, &Hv);
  elaps = (double) (end_time - start_time);
```

FTI : What else ?

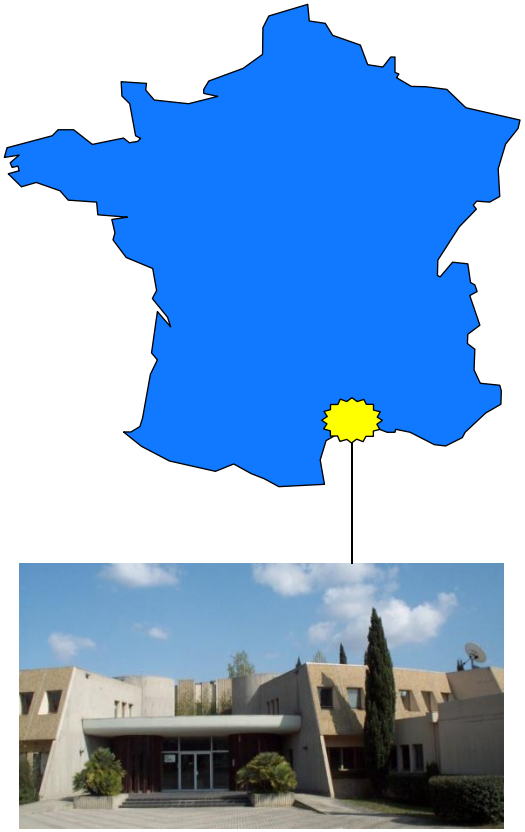
ADIOS ORNL	Multi purpose I/O	Data structure oriented	No restart integrated
FTI ANL	CPR	Data structure oriented	Integrated restart
SCR Scalable C/R LLNL	CPR	File oriented	Integrated restart

Centre Informatique National de l'Enseignement Supérieur

The national computer center of french higher education

CINES
Centre Informatique National de l'Enseignement Supérieur

CINES is located in Montpellier (South of France)



- **CINES is under the supervision of the French Ministry in charge of research.**
- **CINES provides the french public research community with computing resources and services.**
- **50 persons (technicians, engineers and administratives) works in CINES**