FTI: Fault Tolerance Interface

Leonardo Bautista Gomez
Ana Gainaru
Franck Cappello

Fault Tolerance Interface

- **Outline:**
  - Motivations
  - Checkpoint levels
  - Dedicated process
  - Intelligent clustering
  - API + configuration file
  - Traces with VampirTrace
  - Large scale evaluation

Fault Tolerance Interface

- **Motivations:**
  - Fault tolerance is critical at scale.
  - More components can fail.
  - More correlated components.
  - Multiple different types of failures.
  - Power limits might impact reliability

Fault Tolerance Interface

Local Storage: SSD, PCM, NVM.
Fastest checkpoint level.
Low reliability, transient failures.

Partner Copy: Ckpt. Replication.
Fast copy to neighbor node.
It tolerates single node crashes.

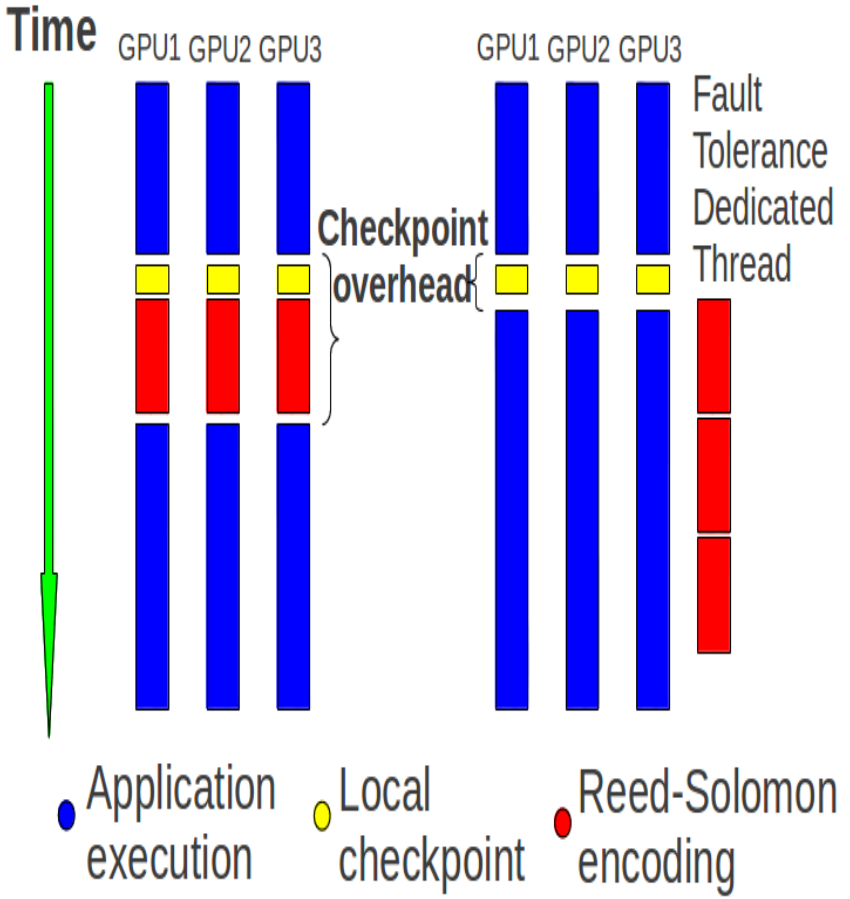
RS Encoding: Ckpt. Encoding.
Slow for large checkppoints.
Very reliable, multiple node crashes.

File System: Classic Ckpt.
Slowest of all levels.
The most reliable. Power outage.

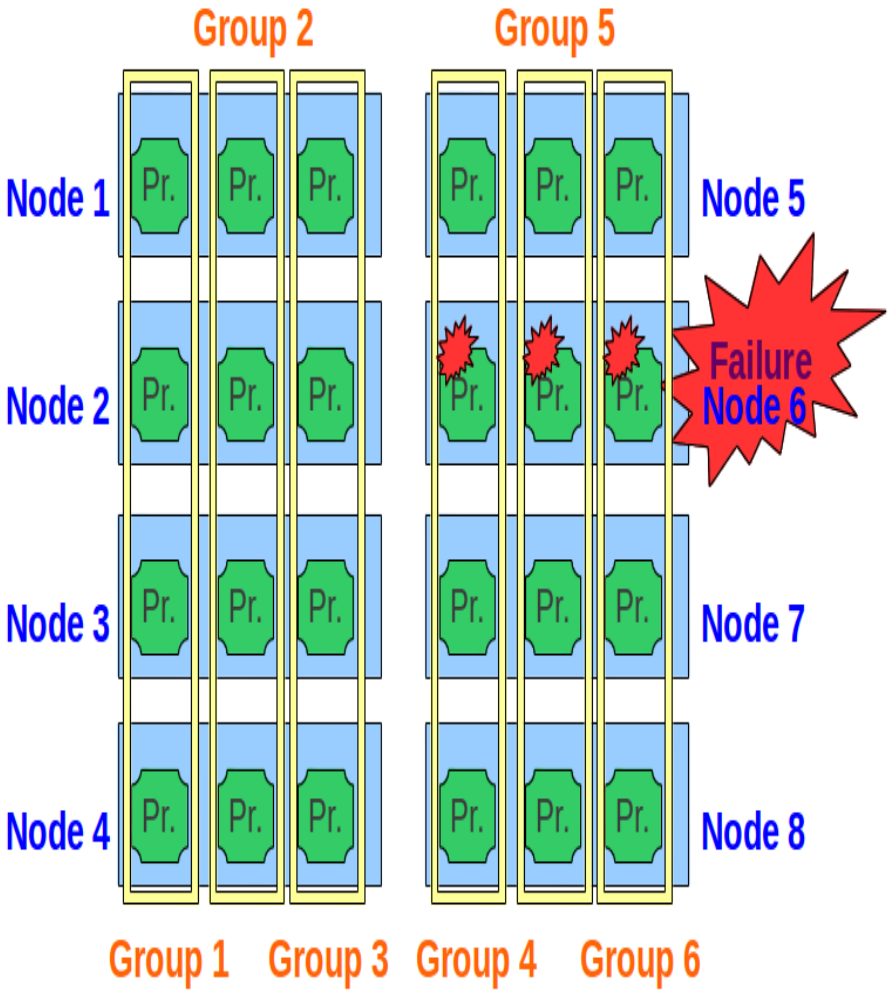
- **Multilevel Ckpt. with several:**
  - Resiliency levels
  - Ckpt. overheads
  - Ckpt. intervals
  - Communications

Fault Tolerance Interface

- **Hybrid systems:**
  - FTI dedicated threads
  - Asynchronous data transfer
  - Reduced ckpt. overhead



Fault Tolerance Interface



- **Topology Aware:**
  - Topology recognition
  - Intelligent clustering
  - Enhanced reliability

Fault Tolerance Interface

- Application-level
- **Simple API:**
  - FTI_Init
  - FTI_Protect
  - FTI_Snapshot
  - FTI_Finalize

```
int main(int argc, char **argv) {

   MPI_Init(&argc, &argv);
   FTI_Init("conf.fti", MPI_COMM_WORLD);

   double *grid;
   int i, steps=500, size=10000;
   FTI_Protect(0, &i,   1,   FTI_INTG);
   FTI_Protect(1, grid, size,FTI_DFLT);

   for (i=0; i<steps; i++) {
      FTI_Snapshot();
      kernel1(grid);
      kernel2(grid);
   }

   FTI_Finalize();
   MPI_Finalize();
   return 0;
}
```

Fault Tolerance Interface

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

Fault Tolerance Interface

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

Fault Tolerance Interface

- **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

Fault Tolerance Interface

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

Fault Tolerance Interface

Configuration file for FTI

```
[basic]
# Set to 1 for having 1 FTI dedicated process per node
Head                            = 1

# Number of processes per node (including FTI dedicated processes)
node_size                       = 2

# Path where local checkpoints will be stored
ckpt_dir                        = /path/to/local/storage/

# Path where global checkpoints will be stored
glbl_dir                        = /path/to/global/storage/

# Path where checkpoints metadata will be stored
meta_dir                        = /path/to/myhome/.fti/

# Checkpoint interval in minutes
ckpt_int                        = 1

# Checkpoint interval for level 2 (in number of L1 checkpoints)
ckpt_l2                         = 2

# Checkpoint interval for level 3 (in number of L1 checkpoints)
ckpt_l3                         = 4

# Checkpoint interval for level 4 (in number of L1 checkpoints)
ckpt_l4                         = 8
```

Fault Tolerance Interface

Configuration file for FTI

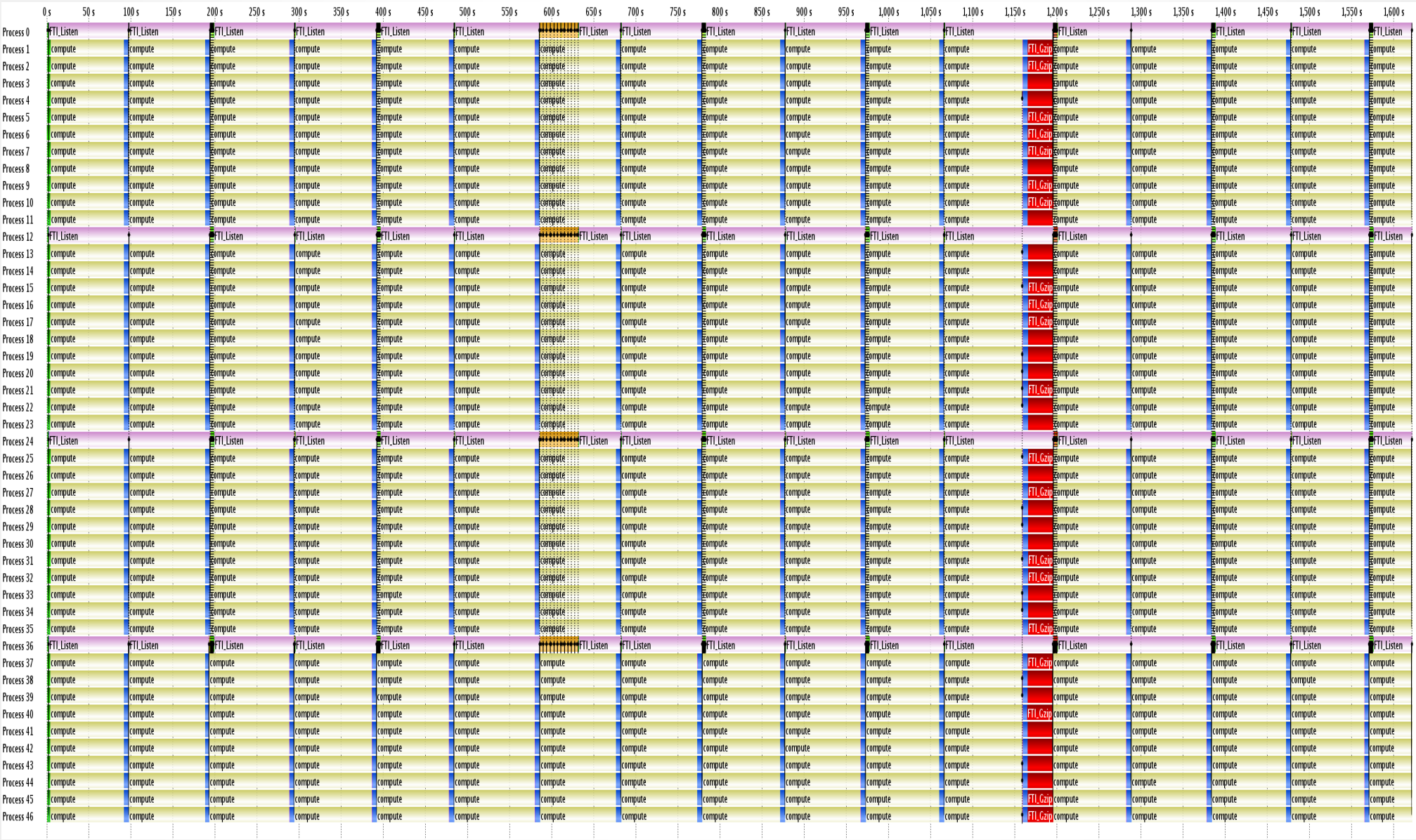
```
[basic]

# Set to 0 to do L2 post-processing asynchronously by the dedicated process
inline_l2                       = 0

# Set to 0 to do L3 post-processing asynchronously by the dedicated process
inline_l3                       = 0

# Set to 0 to do L4 post-processing asynchronously by the dedicated process
inline_l4                       = 0

# Set to 1 to keep the last checkpoint after Finalize
keep_last_ckpt                  = 0

# Size of the group for RS-encoding and Partner-copy ring
group_size                      = 4

# Set to 1 for verbose mode, 2 for moderate, 3 for silent
verbosity                       = 1
```

Fault Tolerance Interface

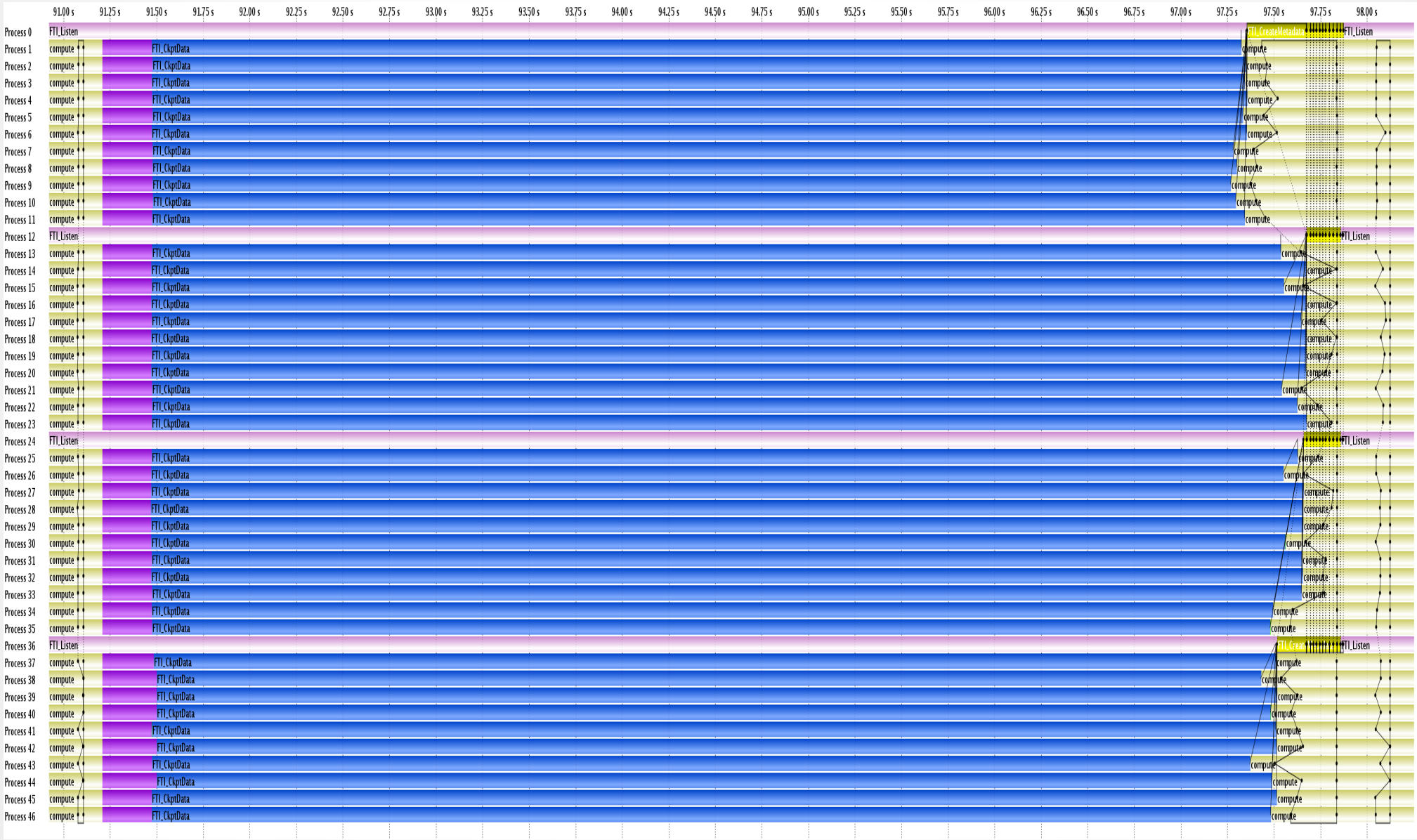
Configuration file for FTI

```
[restart]

# This will be set to 1 automatically after FTI_Init
Failure                         = 0

# This will be set to 1 automatically after FTI_Init
exec_id                         = 2013-11-20_15-01-52


[advanced]

# Block size for communications
block_size                      = 1024

# MPI tag for FTI communications
mpi_tag                         = 2612

# Set to 1 for local tests in one single node (for developers)
local_test                      = 1
```

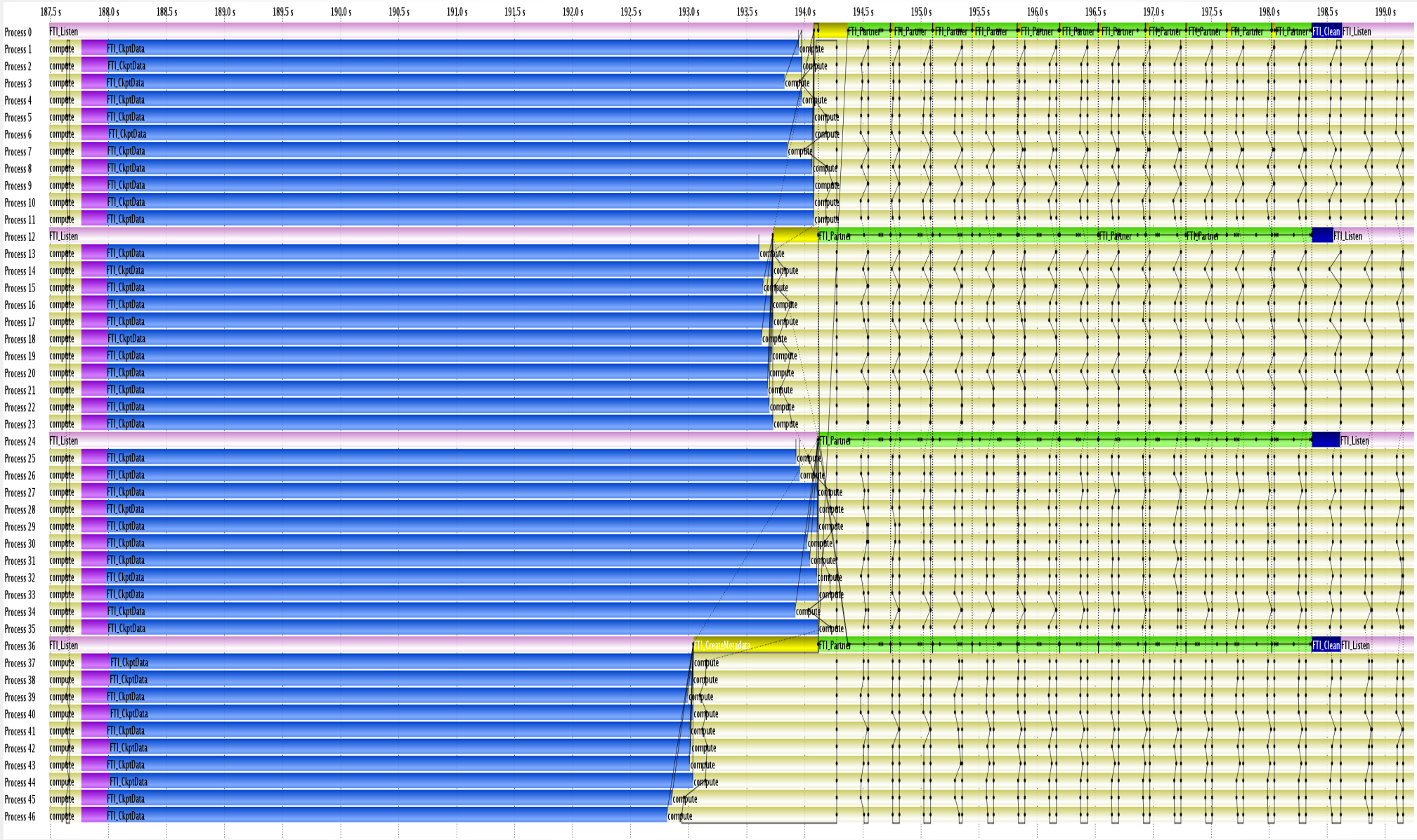
Fault Tolerance Interface



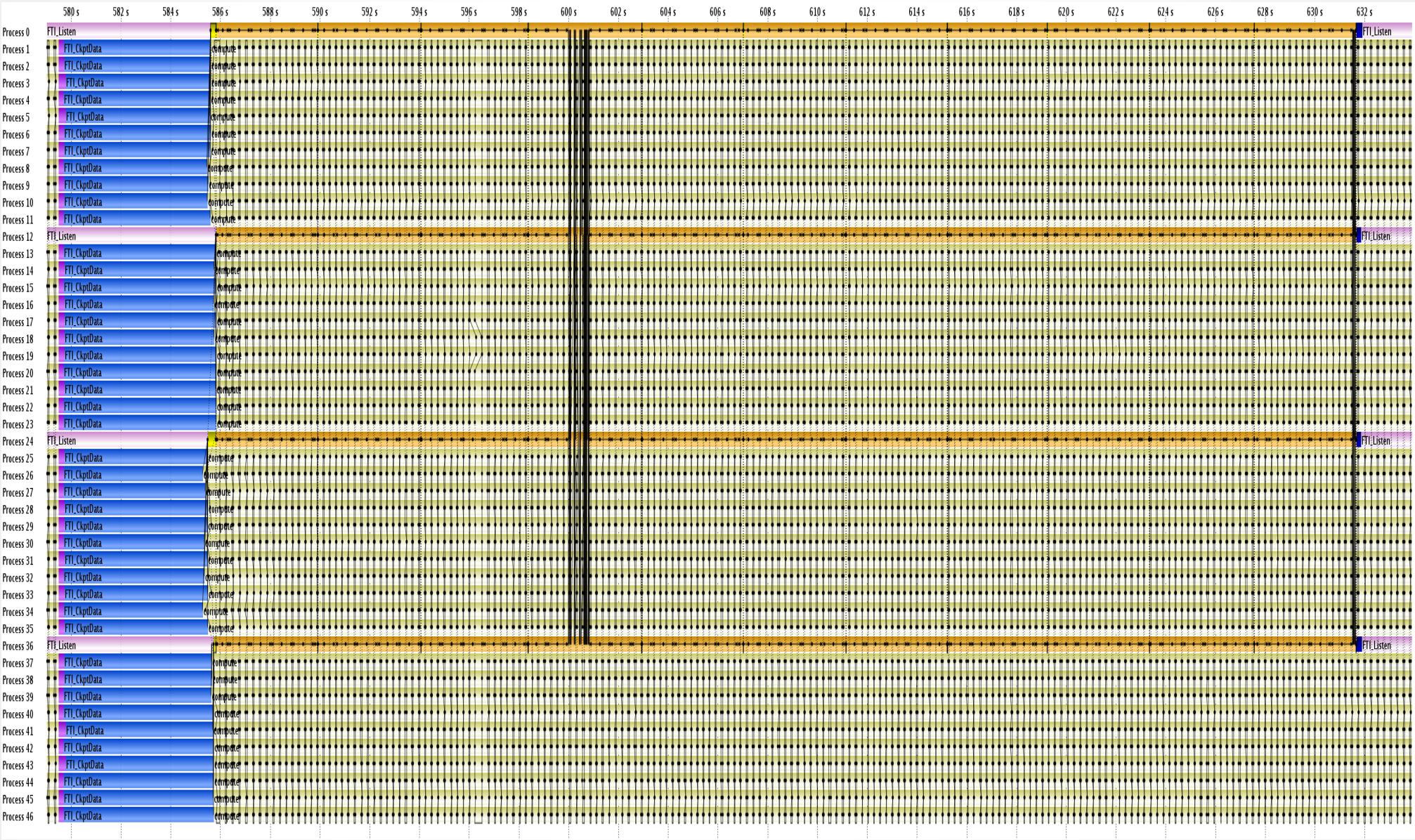
Fault Tolerance Interface



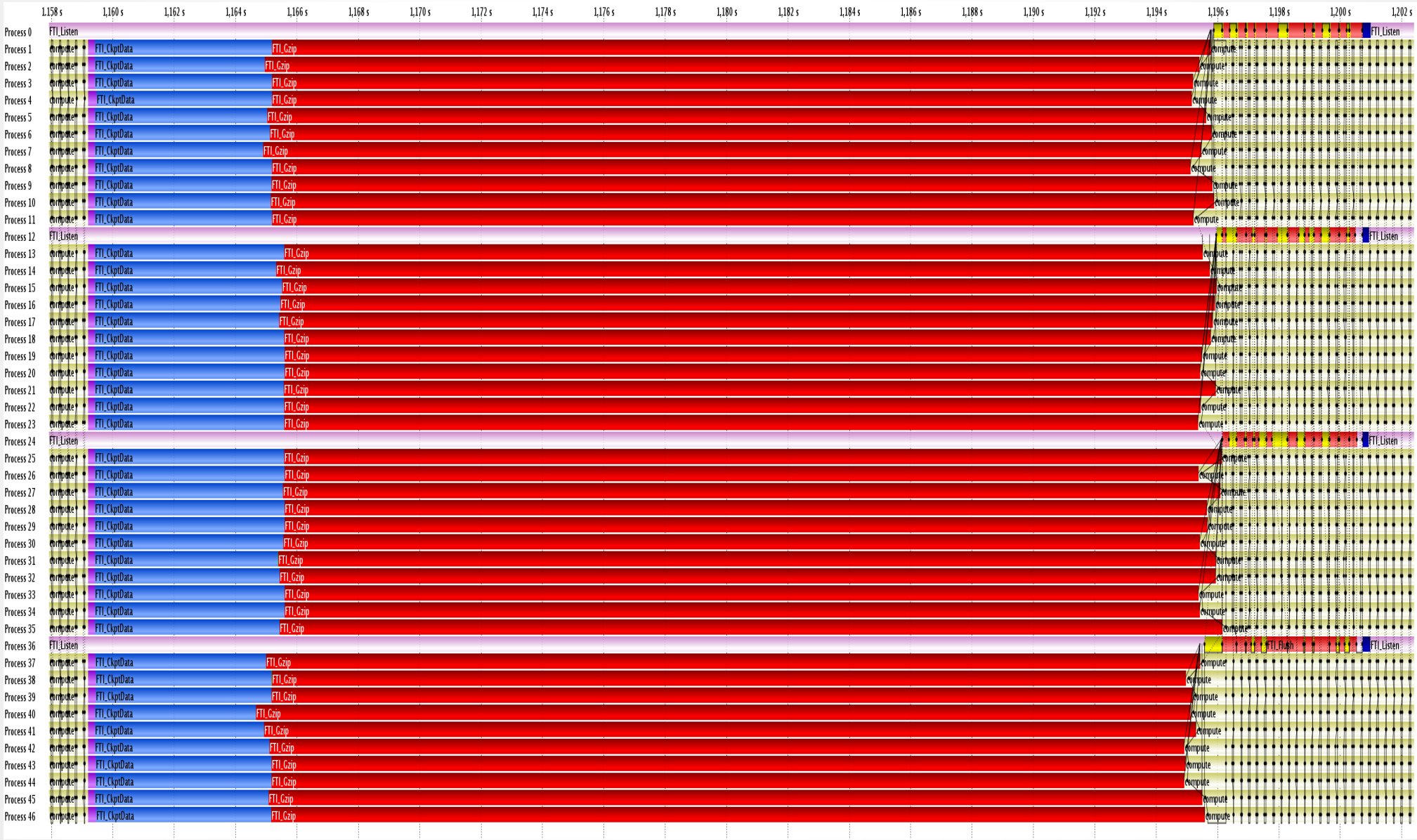
Fault Tolerance Interface



Fault Tolerance Interface



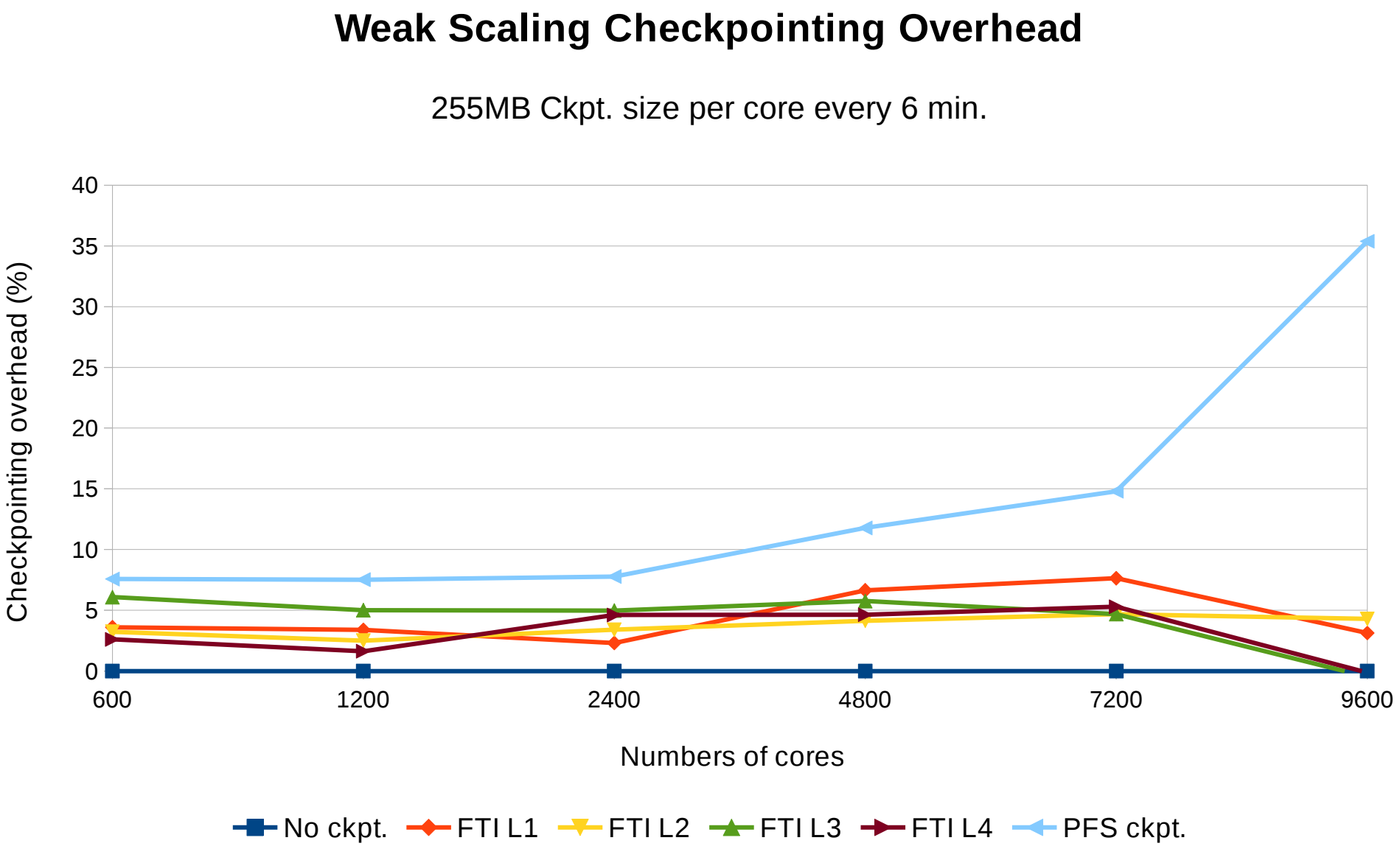
Fault Tolerance Interface



Fault Tolerance Interface

- **Large scale evaluation:**
  - CURIE supercomputer in France
  - SSD on the compute nodes (16 cores)
  - HYDRO scientific application
  - Using 1 FTI dedicated process per node
  - Checkpointing every ~6 minutes
  - Weak scaling to almost 10k processes

Fault Tolerance Interface



Fault Tolerance Interface

- Multilevel checkpointing for HPC
- Open Source under LGPL
- *http://leobago.com/projects/fti/*
- *leobago@mcs.anl.gov*

Fault Tolerance Interface

Thank you!

Questions?

leobago@mcs.anl.gov