Coupling failure prediction, proactive and preventive checkpoint for current production HPC systems.

Ana Gainaru, Leonardo Bautista-Gomez,
Franck Cappello

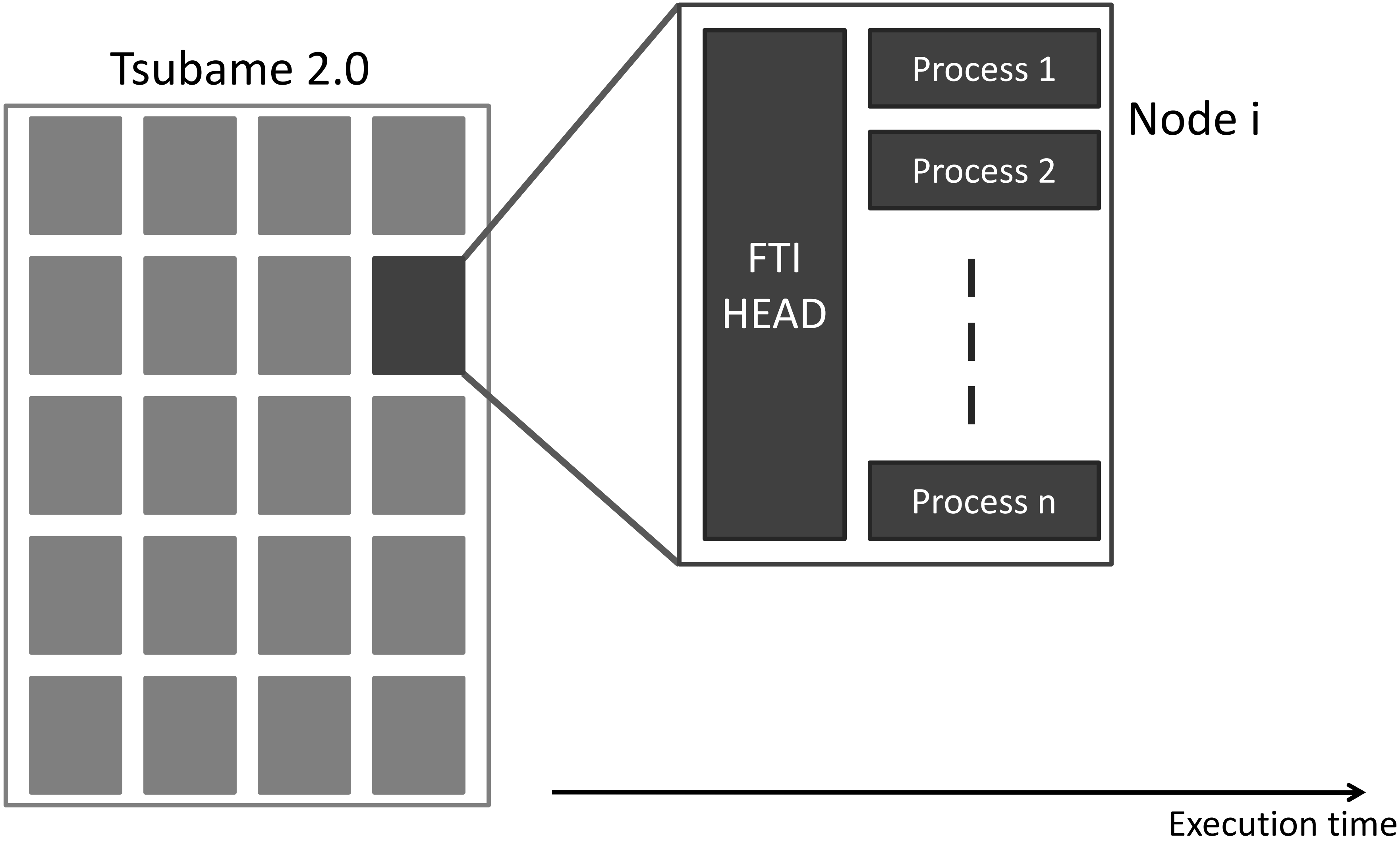
Joint Laboratory for Petascale Computation

Motivation

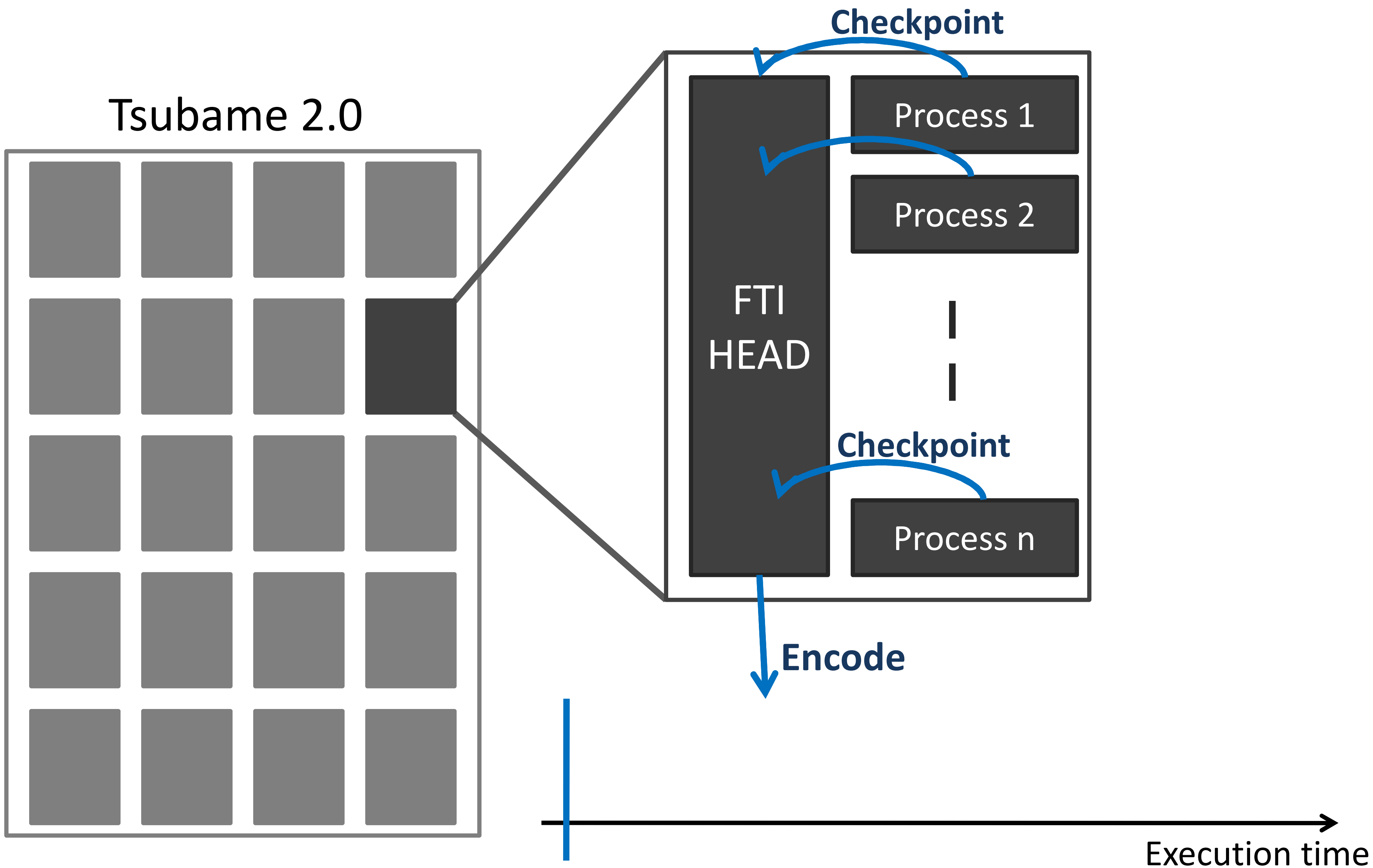
- Prediction is feasible
  - ELSA: Signal analysis with data mining
    - 90% precision and 45% recall
    - At least 10 seconds delay
- Fast checkpointing strategies exist
  - FTI (Fault Tolerance Interface)
    - Capable of taking a checkpoint in ~5s for 1GB memory
    - Multi-level checkpoint with 8% overhead
- Plan
  - Merging FTI with ELSA

Table of contents

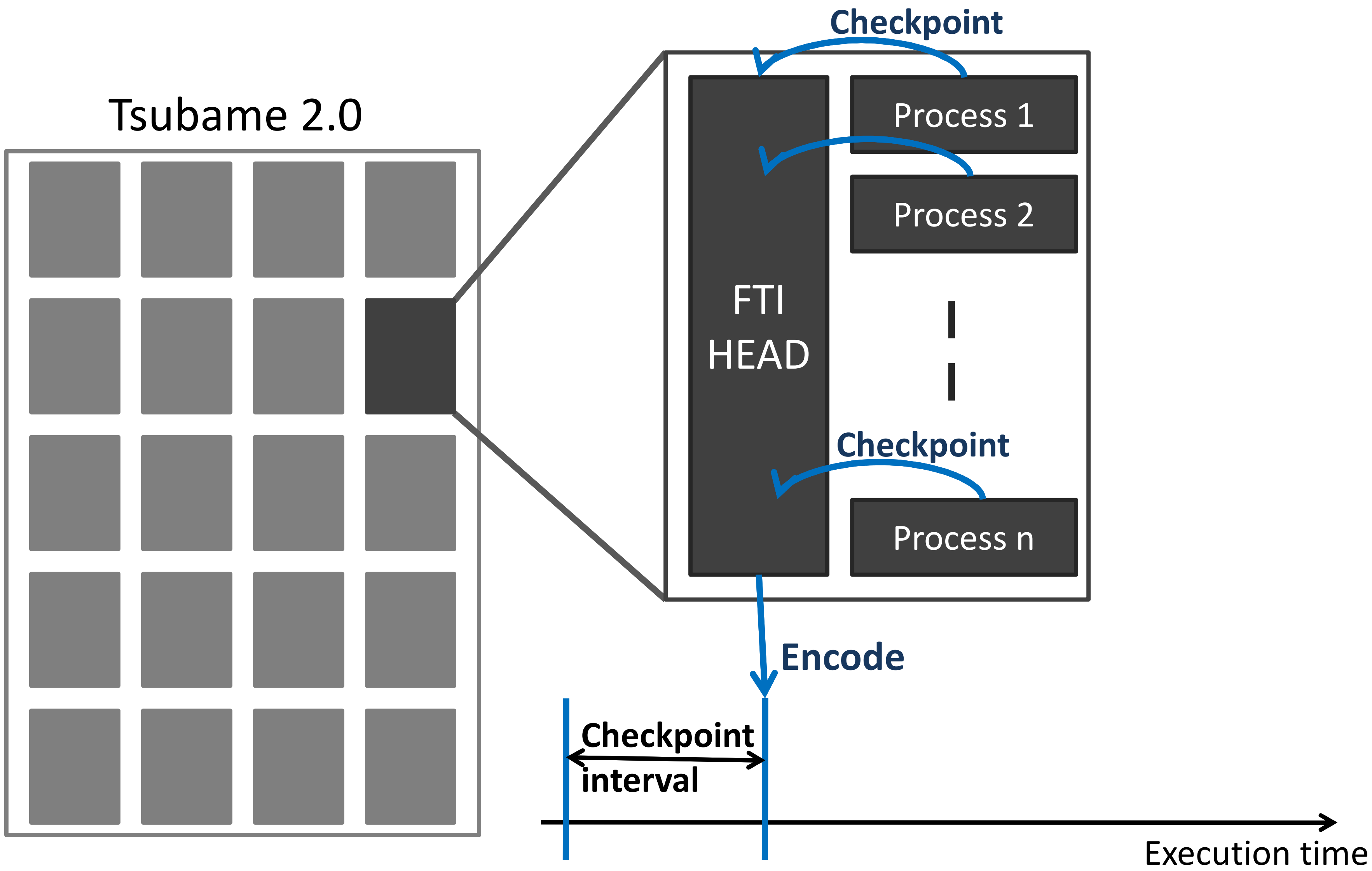
Let's remember FTI



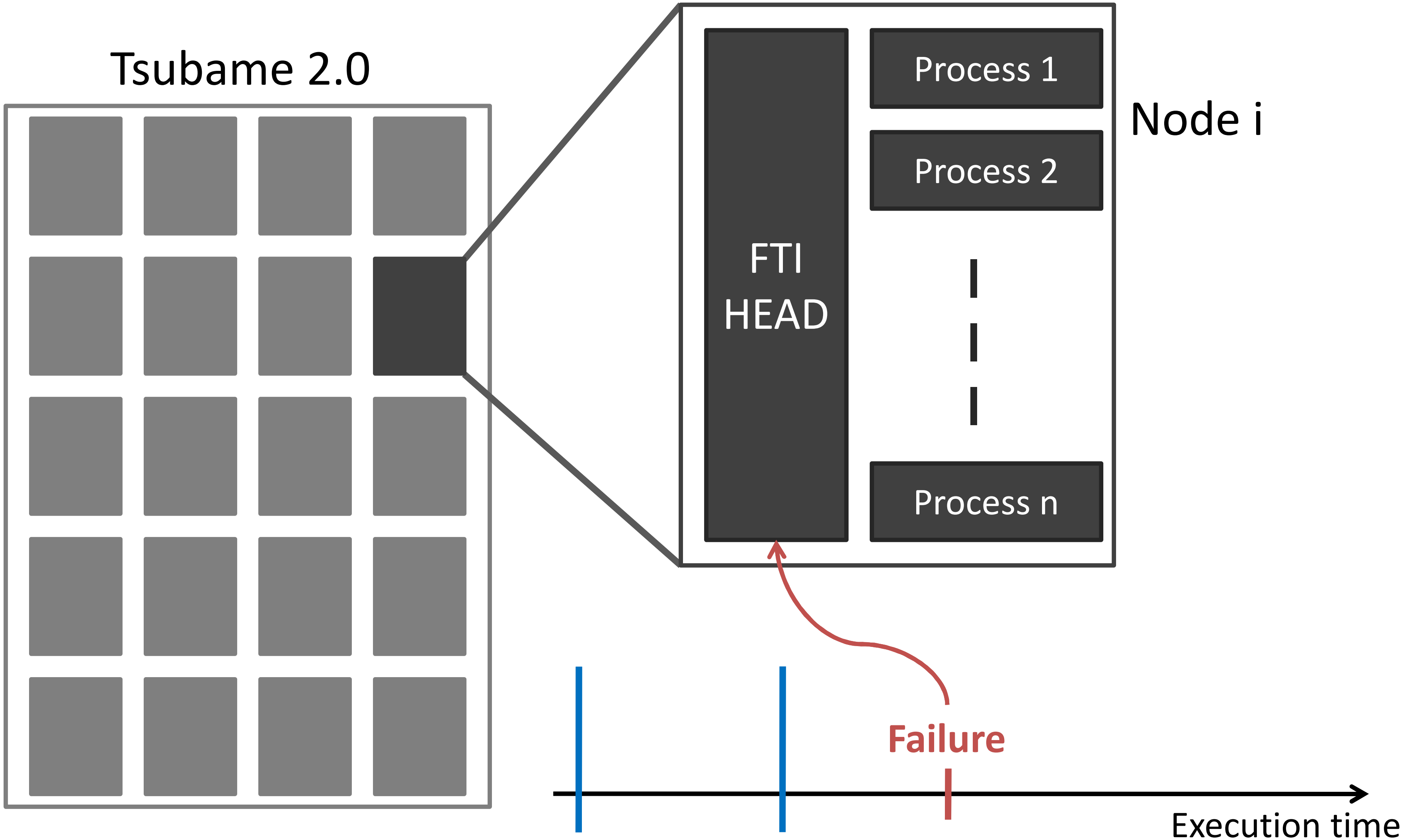
Let’s remember FTI



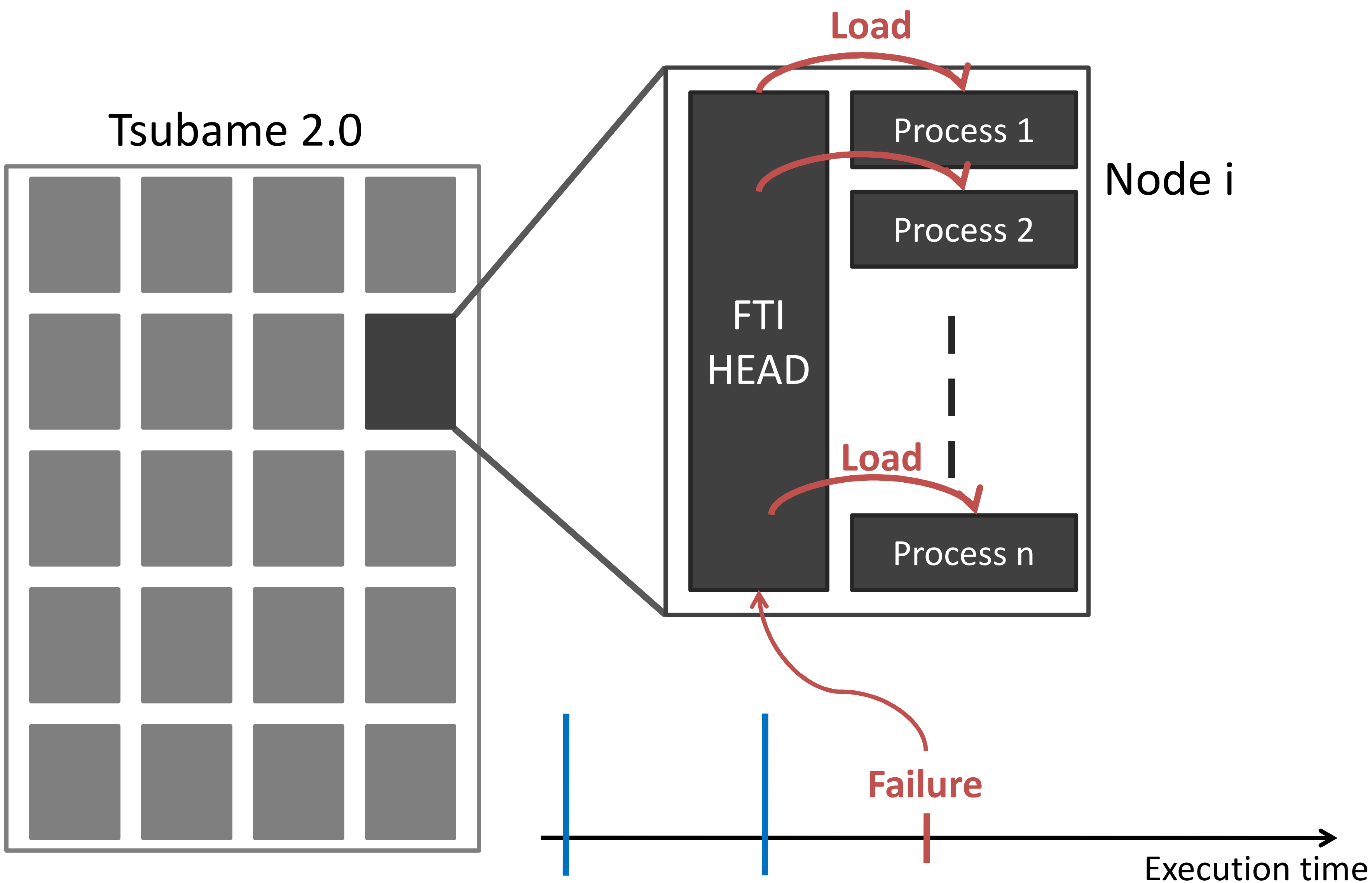
Let’s remember FTI



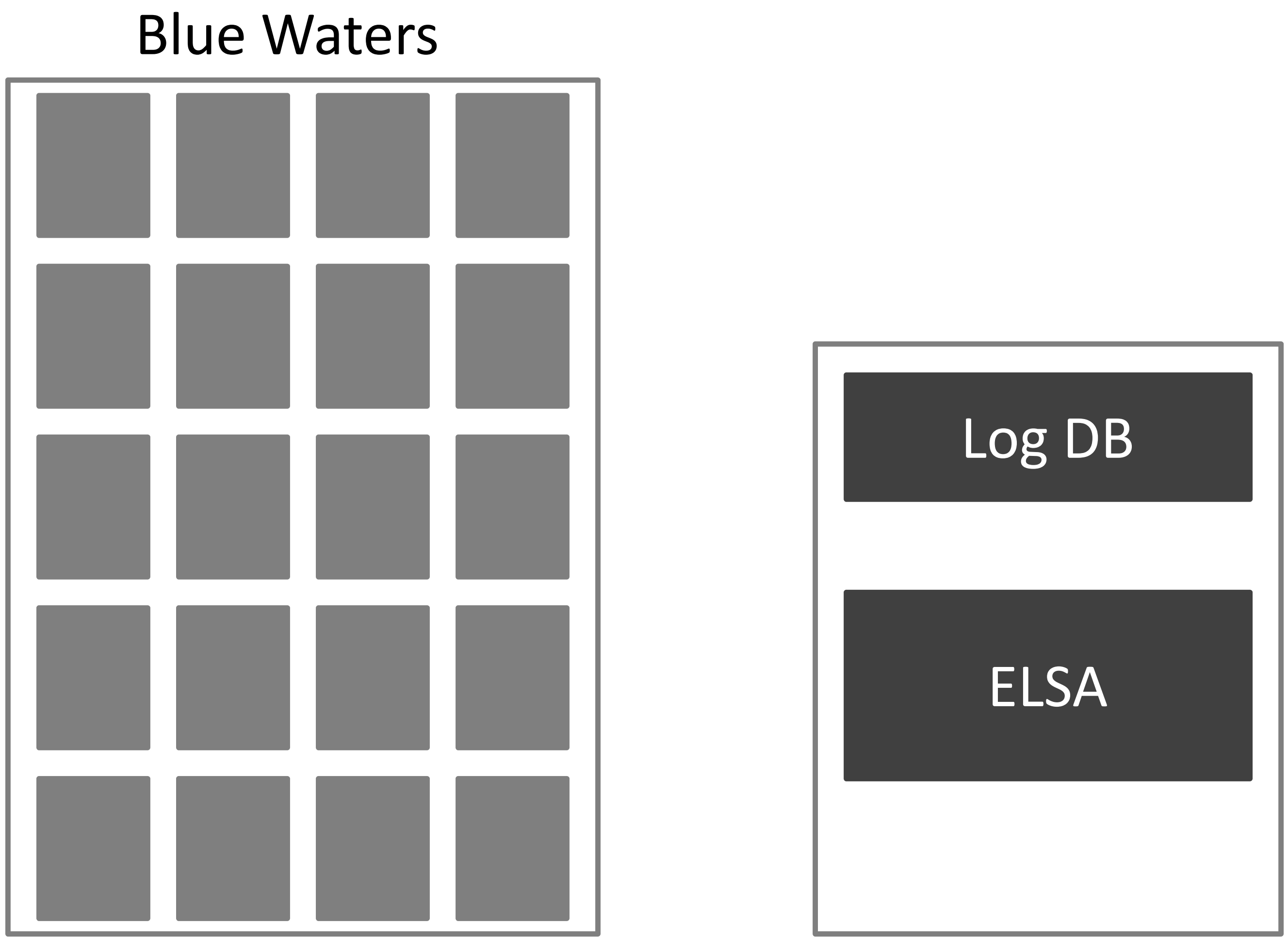
Let's remember FTI



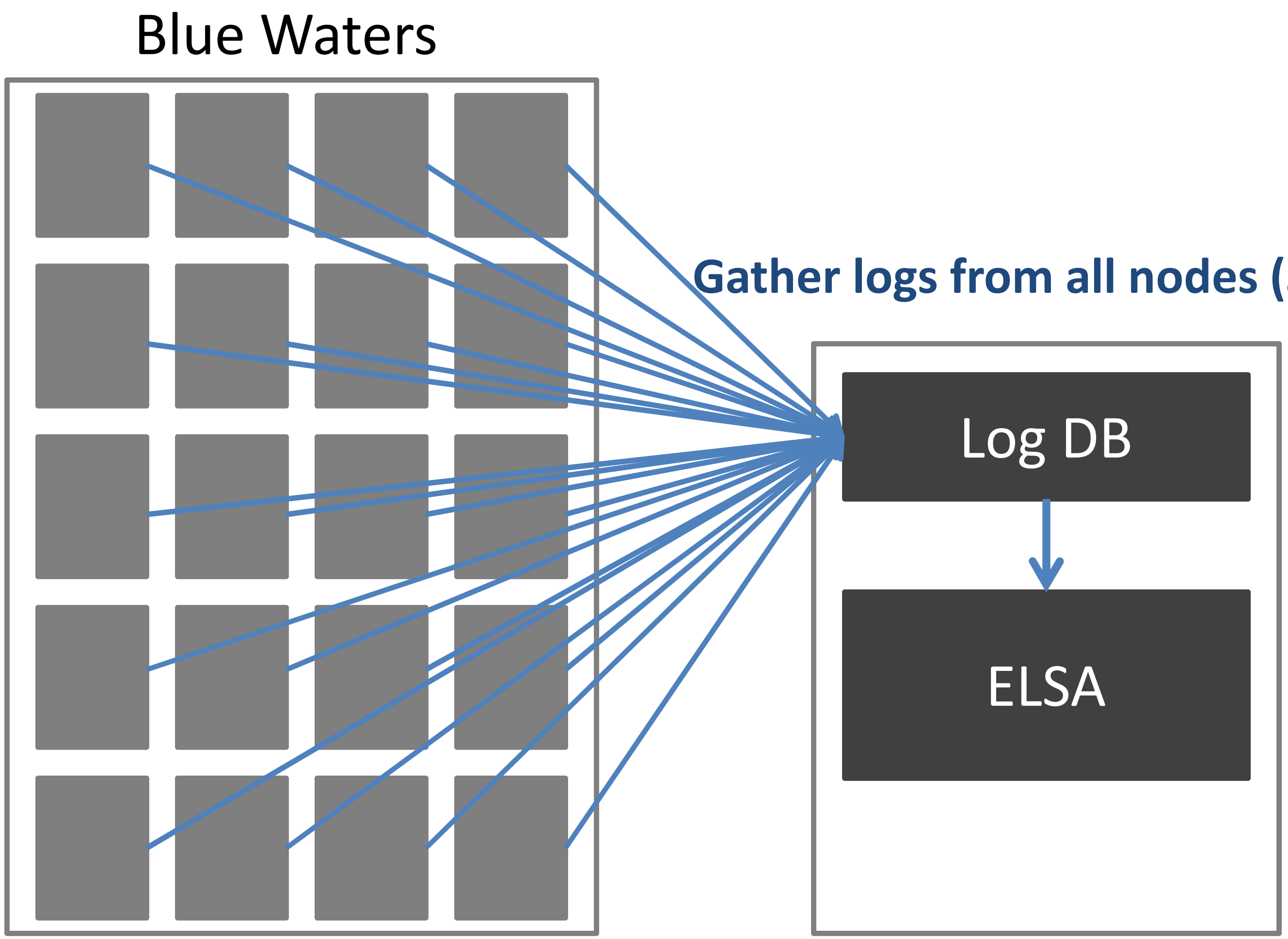
Let's remember FTI



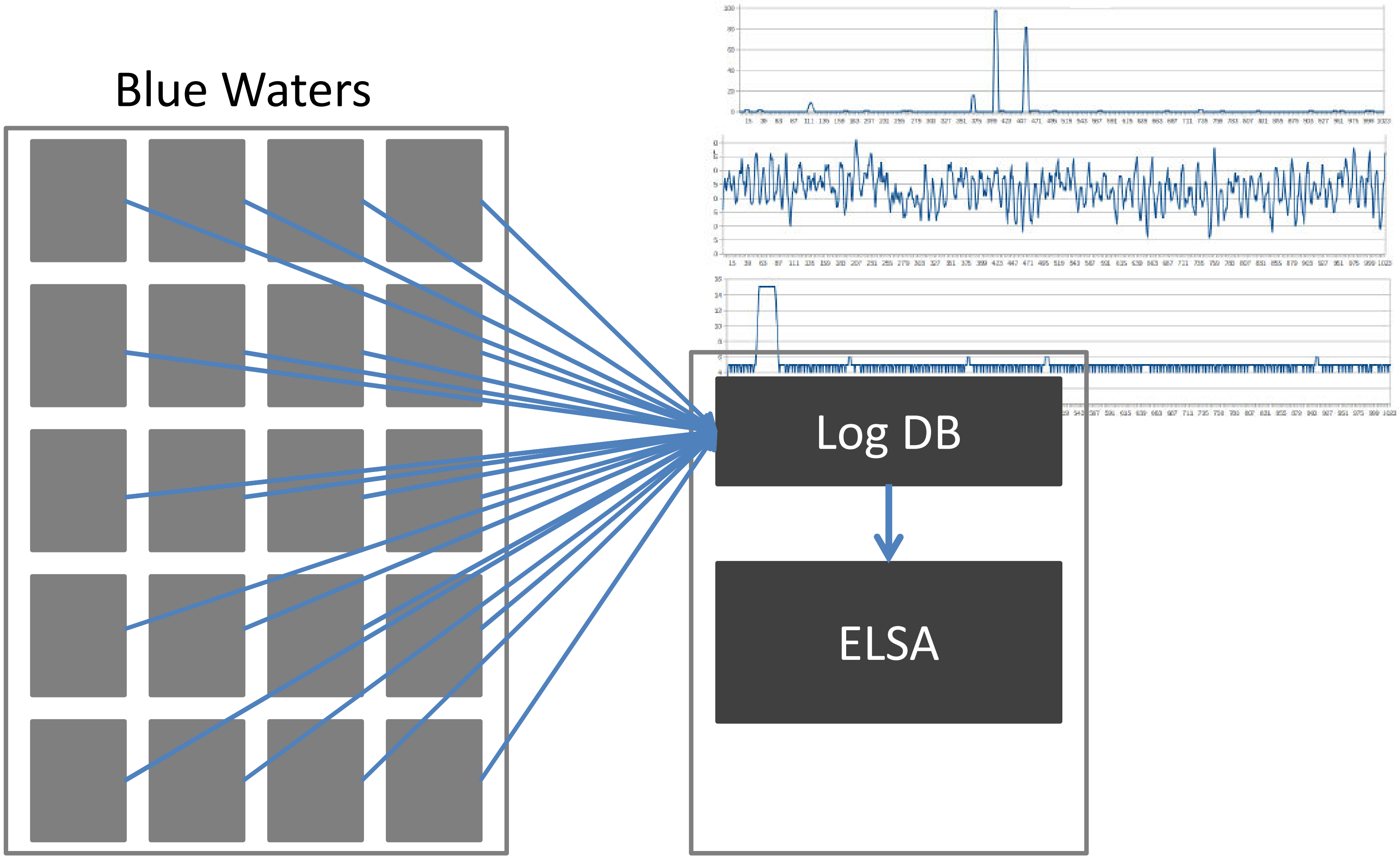
Let’s remember ELSA



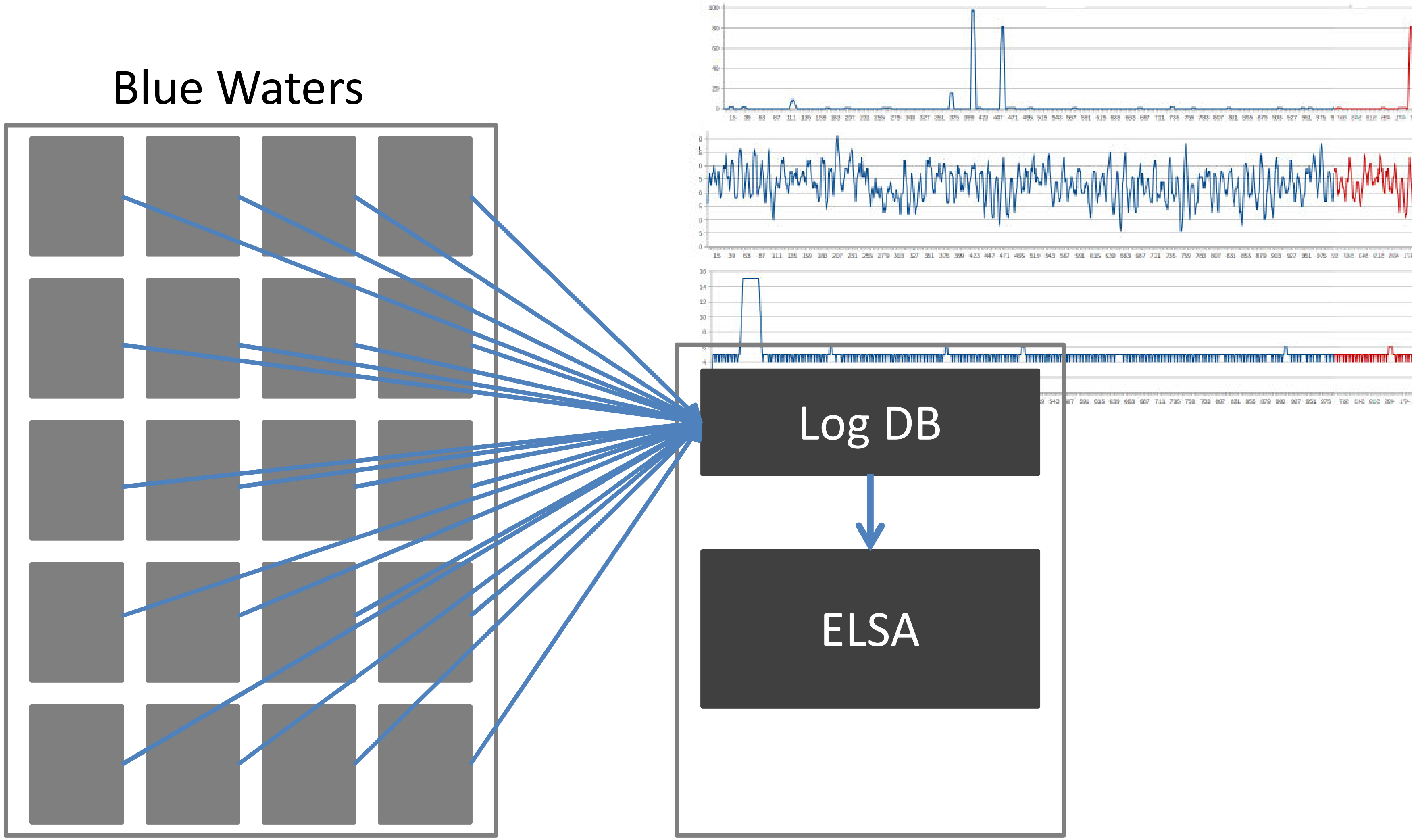
Let's remember ELSA



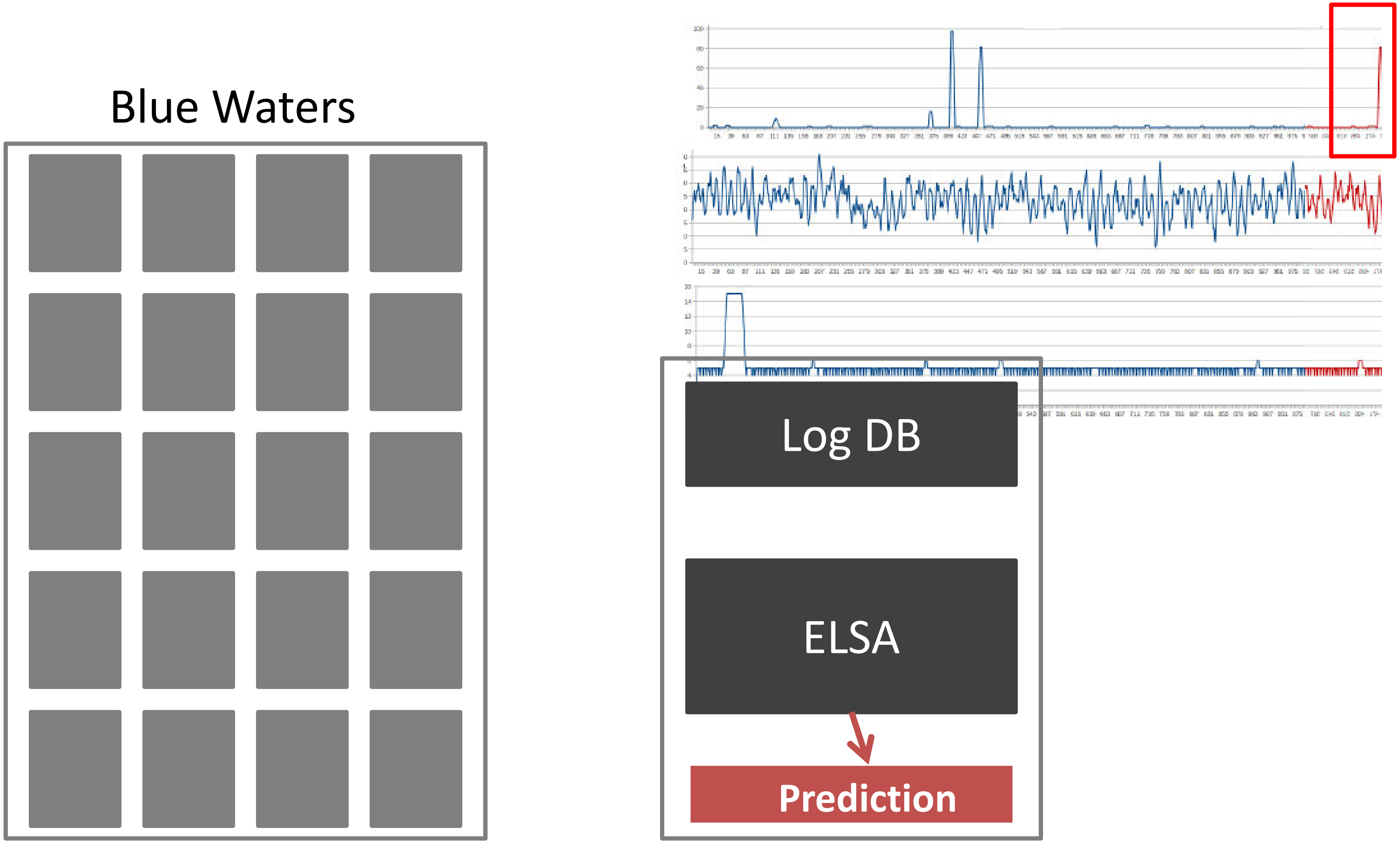
Let's remember ELSA



Let’s remember ELSA



Let's remember ELSA



Let's remember ELSA

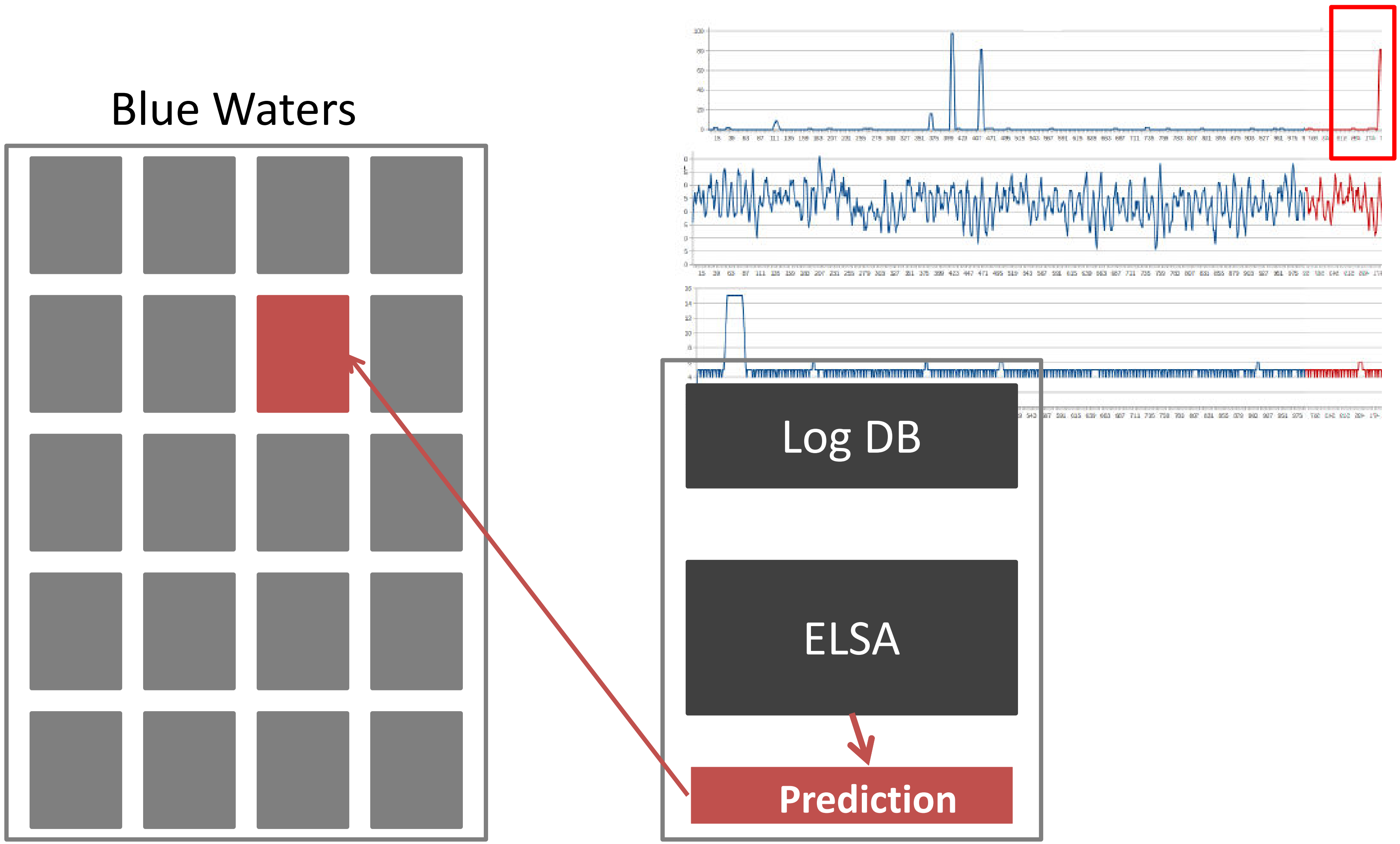
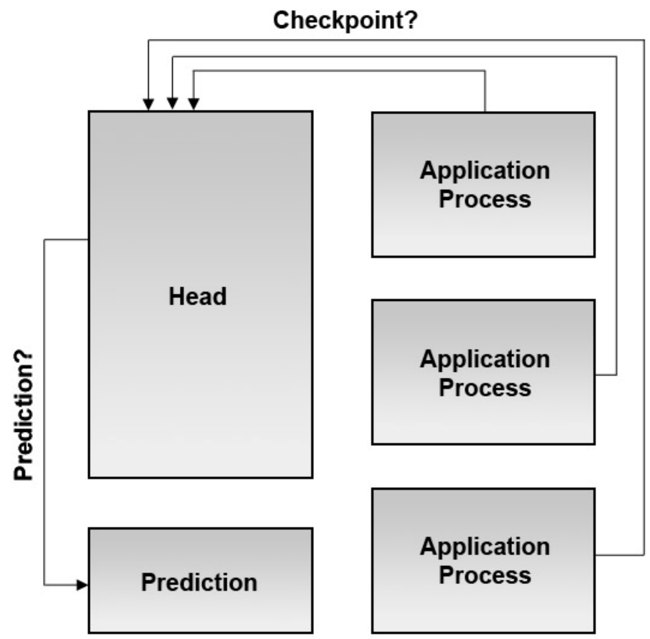


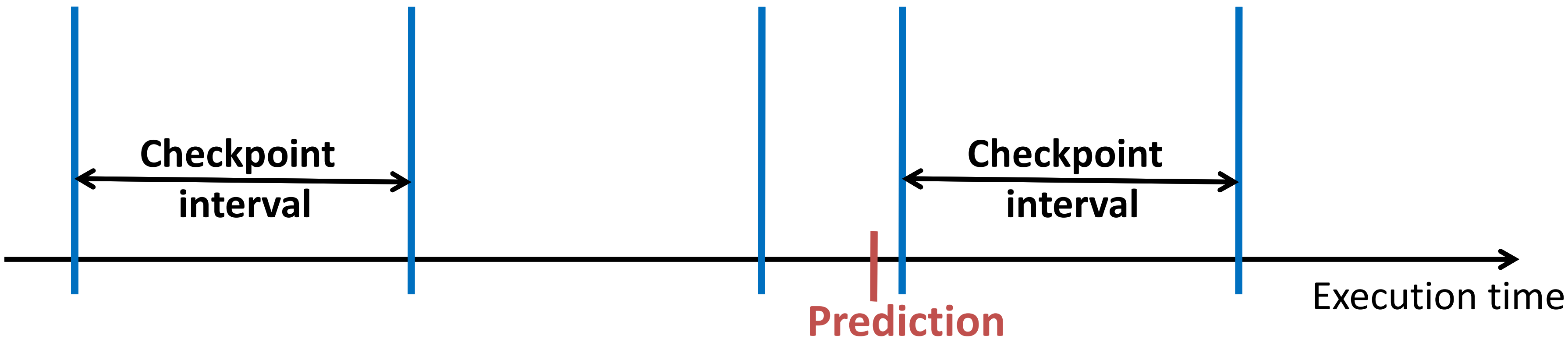
Table of contents

Plan

- Solution:
  - Include ELSA in the Head process
  - Every x seconds the Head checks predictions
  - The Head requests a checkpoint from the app processes
- 3 possible methodologies:
  - Fixed checkpoint interval:
    - Set w/o considering prediction
    - Resets after each prediction
    - Established by the recall value
  - Additional checkpoints for each prediction

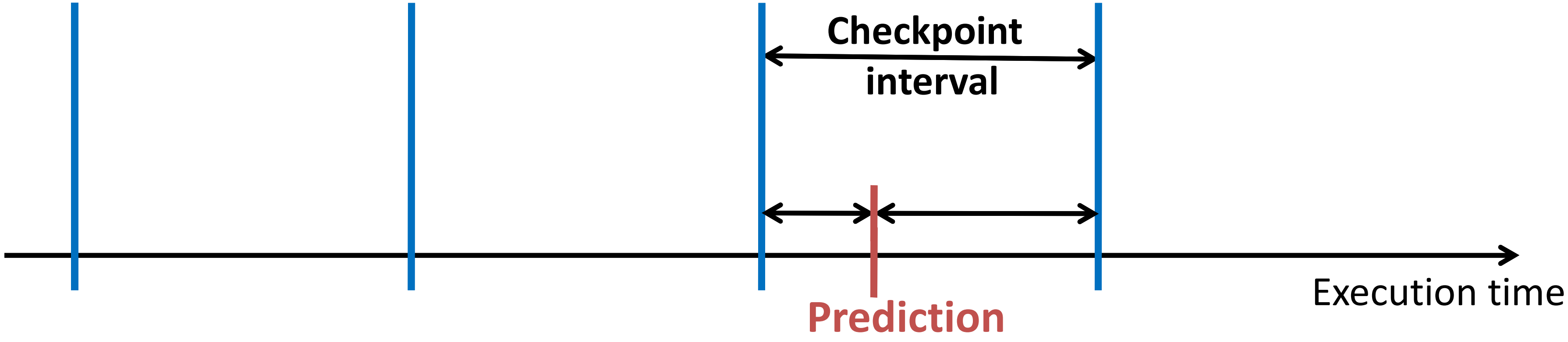


Possible model



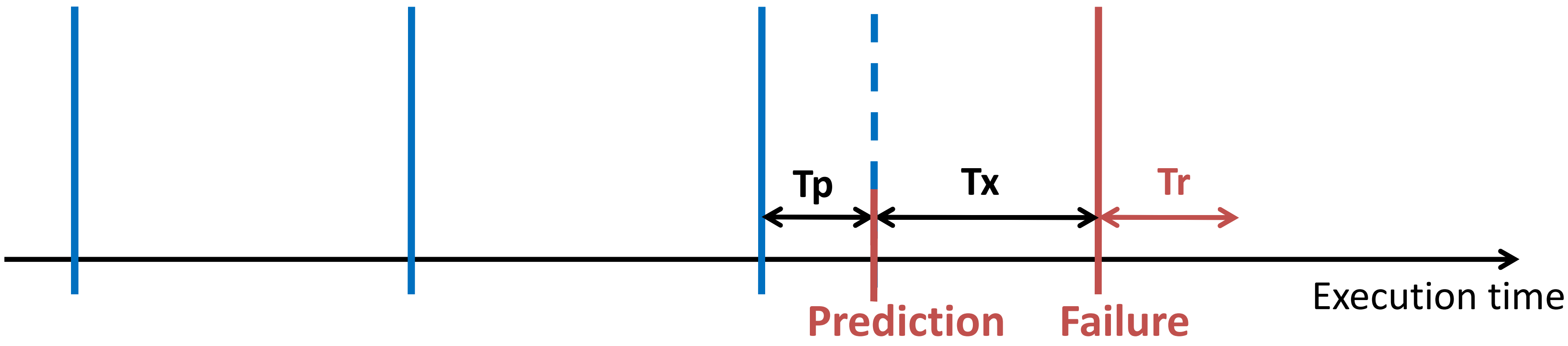
- Checkpoint interval:
  - Given by the MTBF and the checkpoint time
- Prediction:
  - Recall gives the MTB false negative F
    - Depending on the distribution of the failures after prediction

Possible model



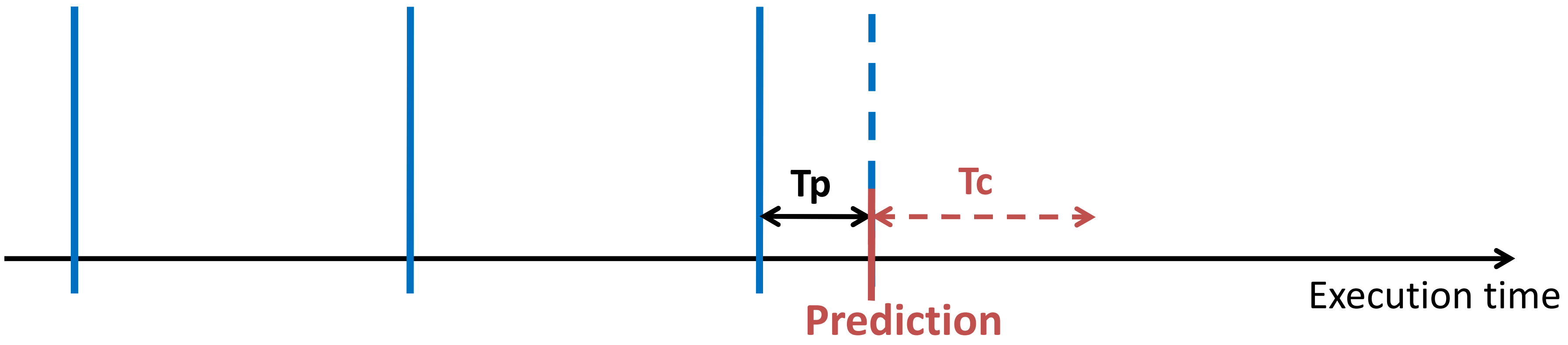
- Depending on the moment of the prediction
  - Decide to take or not a checkpoint
  - Analytical model for 2 cases:
    - Decide to take an extra checkpoint due to the prediction
    - Do not take a checkpoint and just leave the failure to occur without doing any action

Possible model



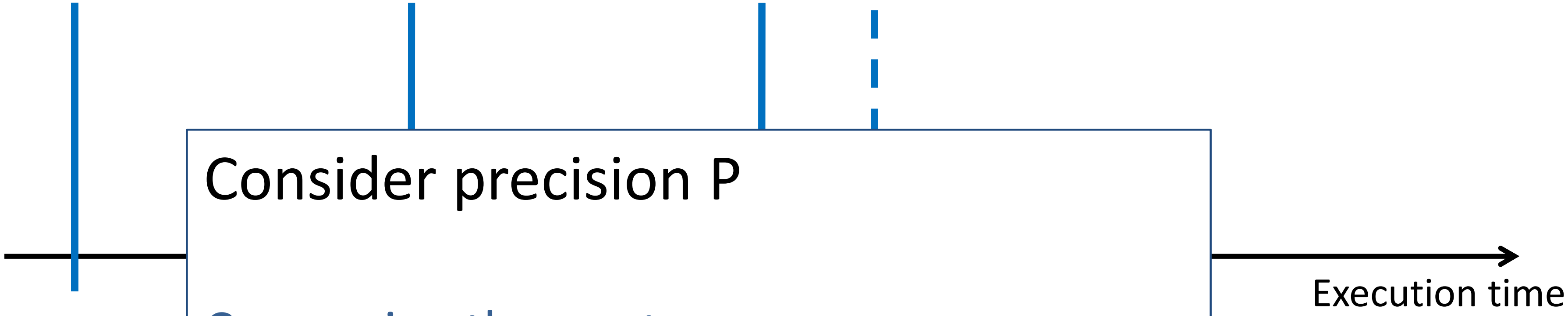
- First case: Prediction is correct
  - Do not checkpoint
    - Waste = Tp + Tx + Tr
  - Take a checkpoint
    - Waste = Tx + Tr

Possible model



- First case: Prediction is wrong
  - Do not checkpoint
    - Waste = 0
  - Take a checkpoint
    - Waste = Tc

Possible model



Consider precision P

Comparing the wastes:

- If $T_p < (1-P)/P * T_c$ - not worth taking a checkpoint

- Firs
  – D
    - Waste = 0
  – Take a checkpoint
    - Waste = Tc

Table of contents

Modifications to FTI

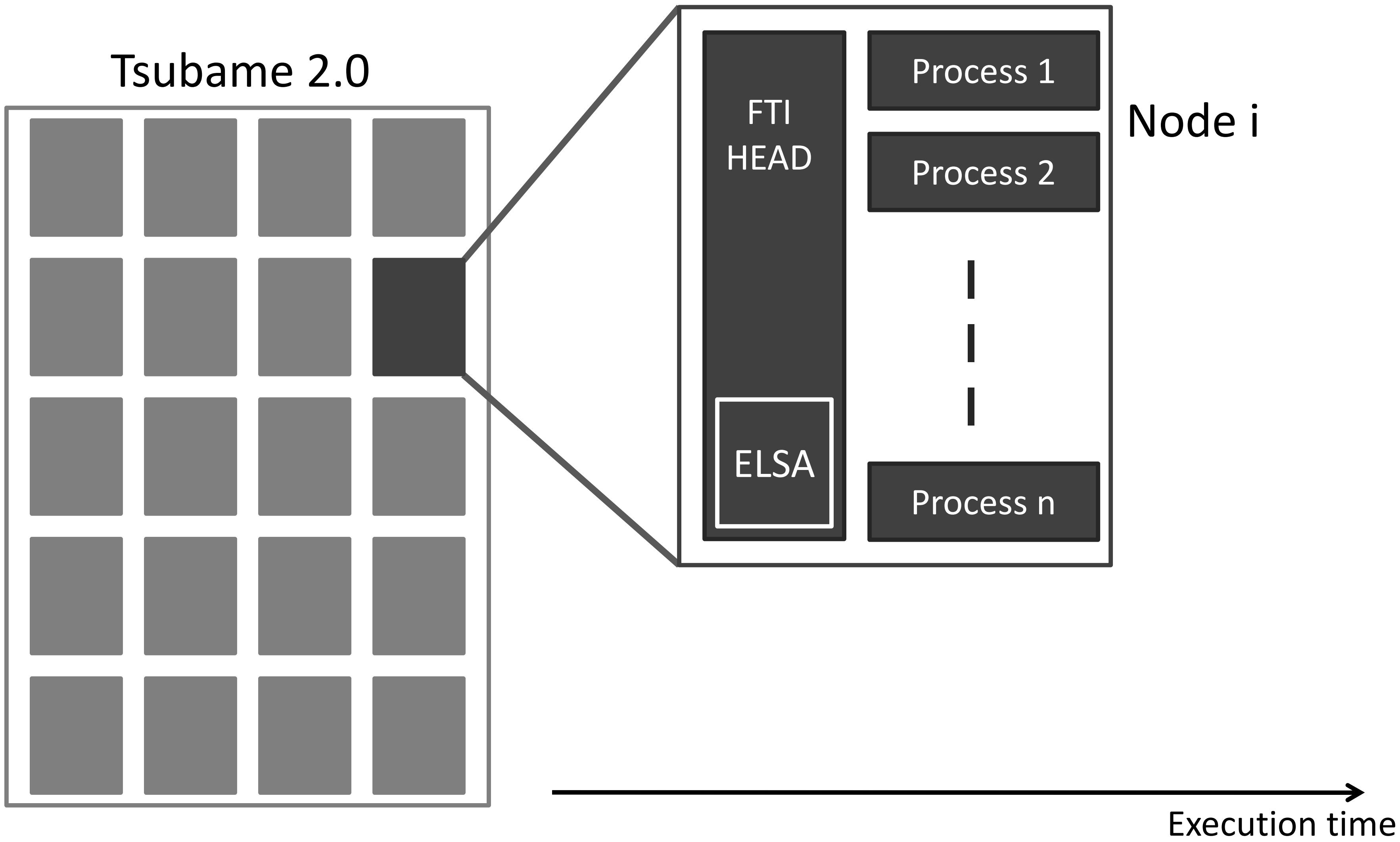
- 2 main contributions:
  - Communication between the Head and application processes
  - Checking predictions regularly

Modifications to FTI

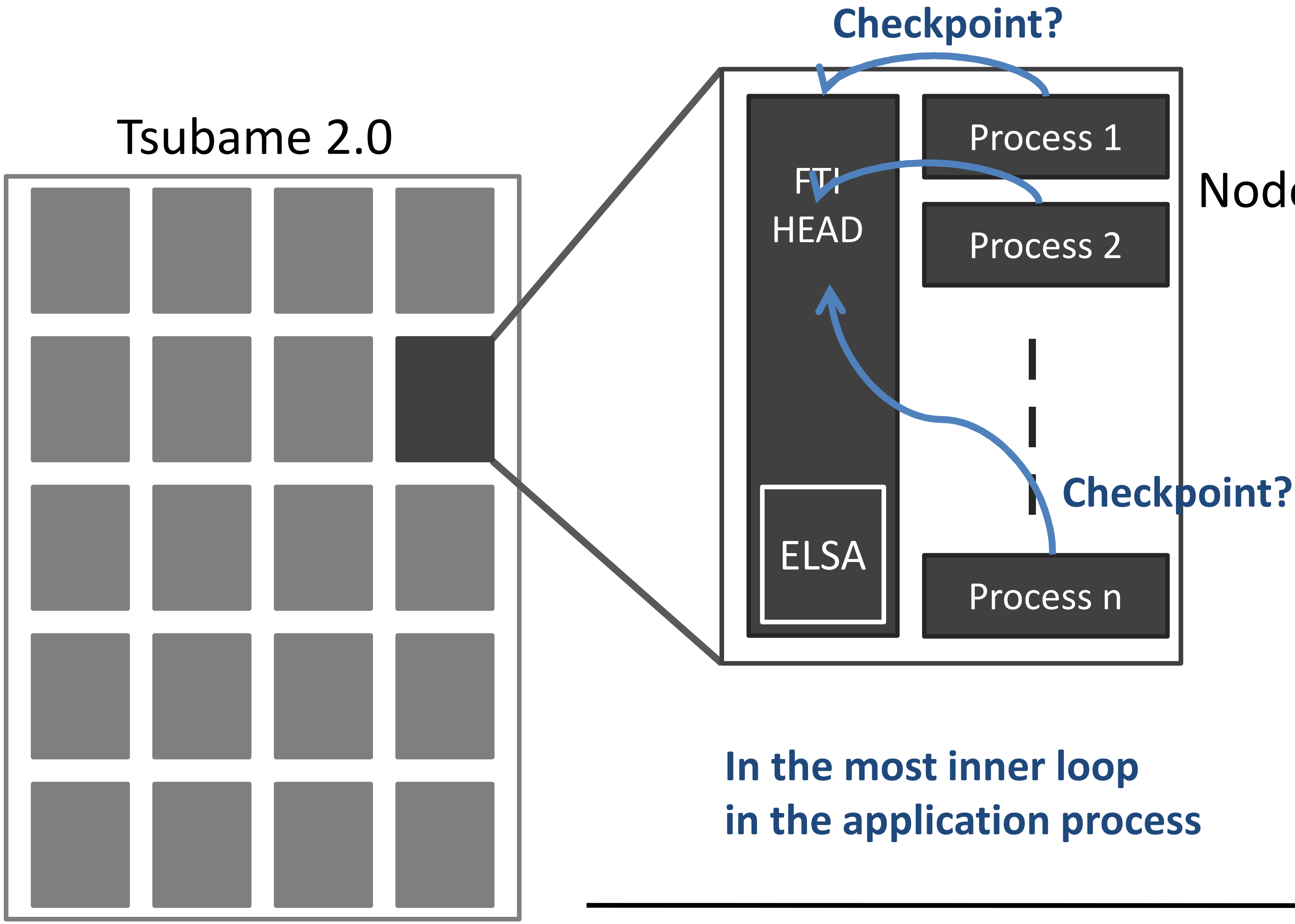
- 2 main contributions:
  - **Communication between the Head and application processes**
  - Checking predictions regularly

- Communication
  - On prediction the Head must force the app processes to checkpoint

Modifications to FTI

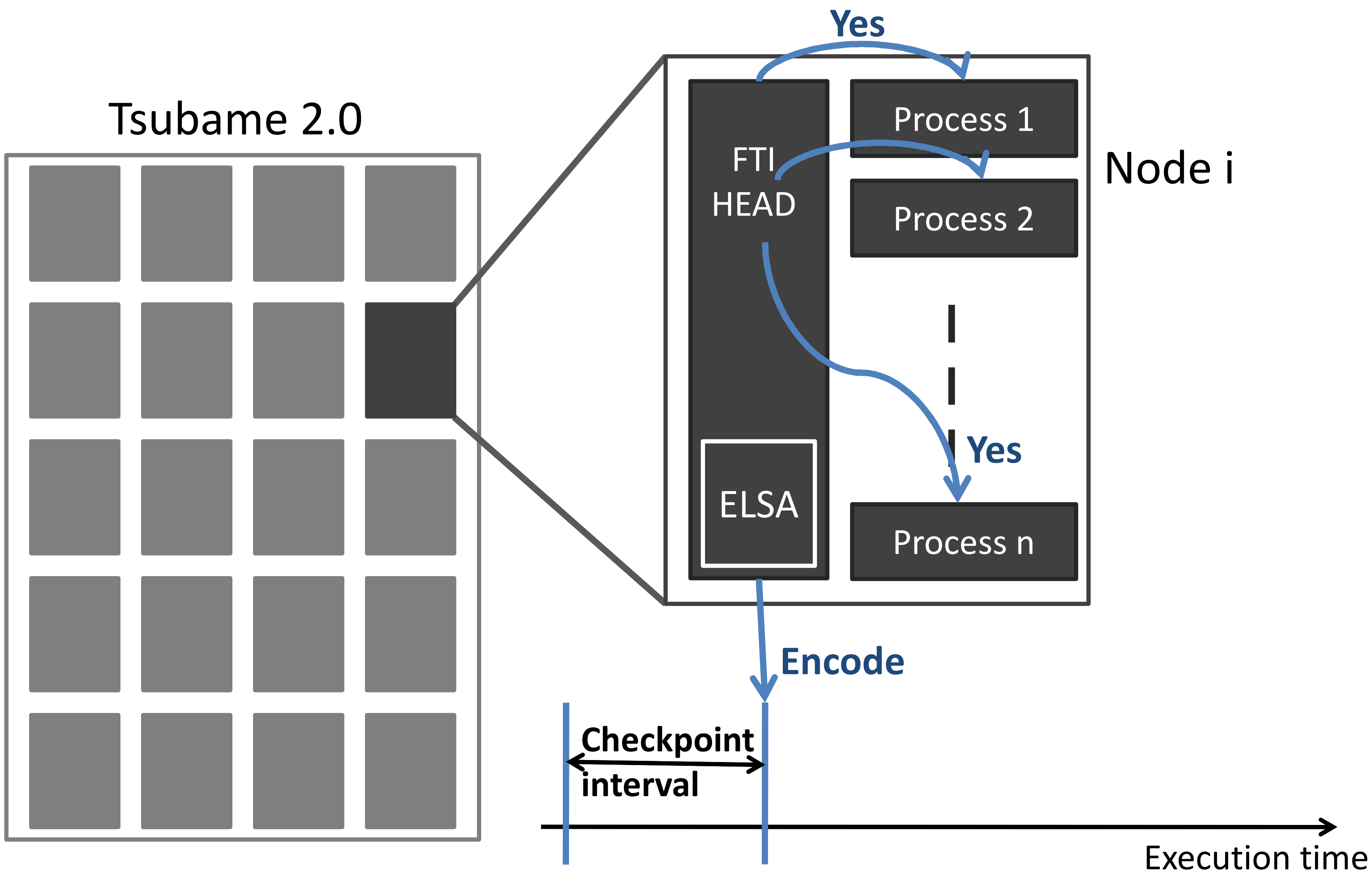


Modifications to FTI



In the most inner loop
in the application process

Modifications to FTI

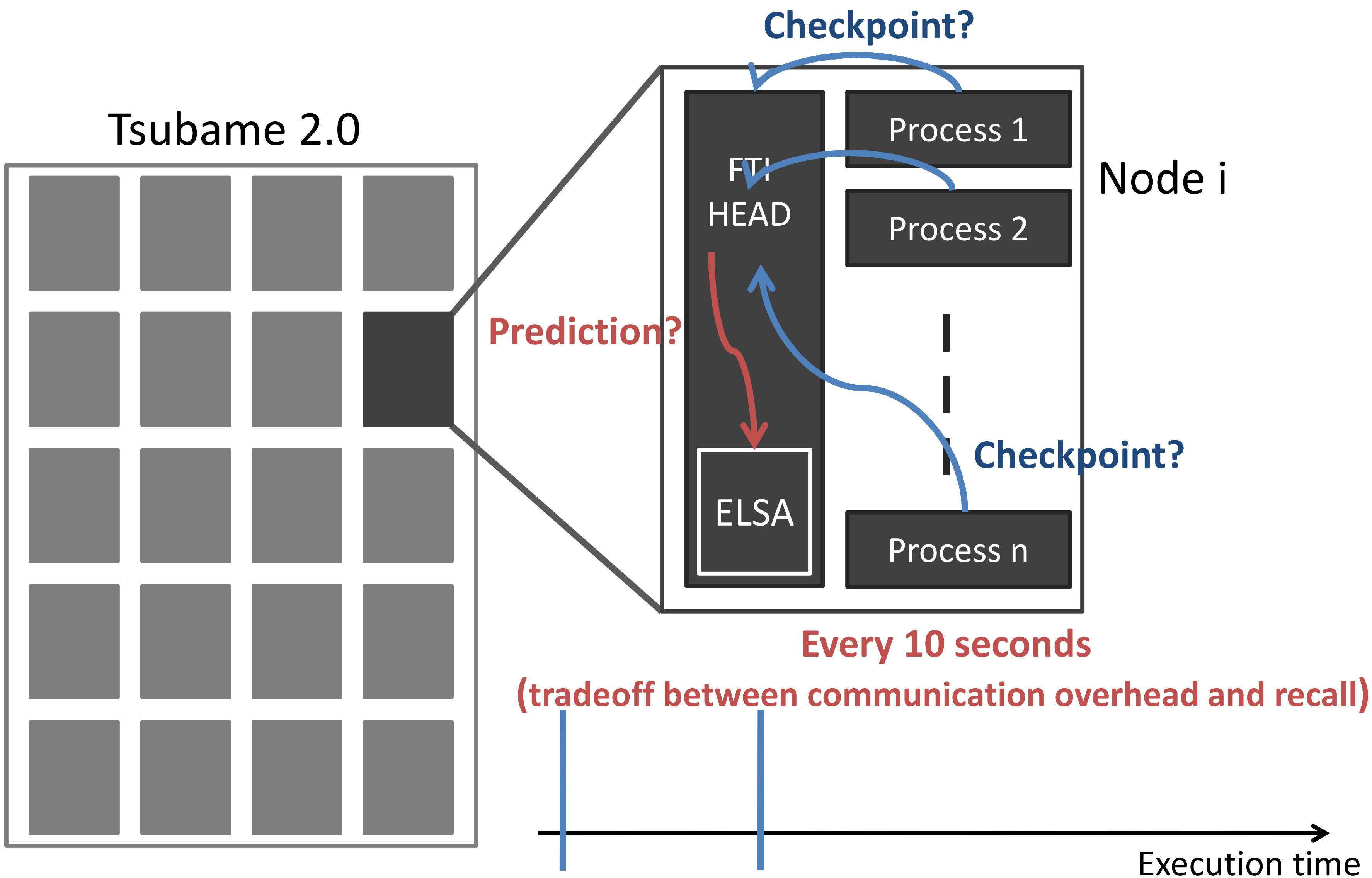


Modifications to FTI

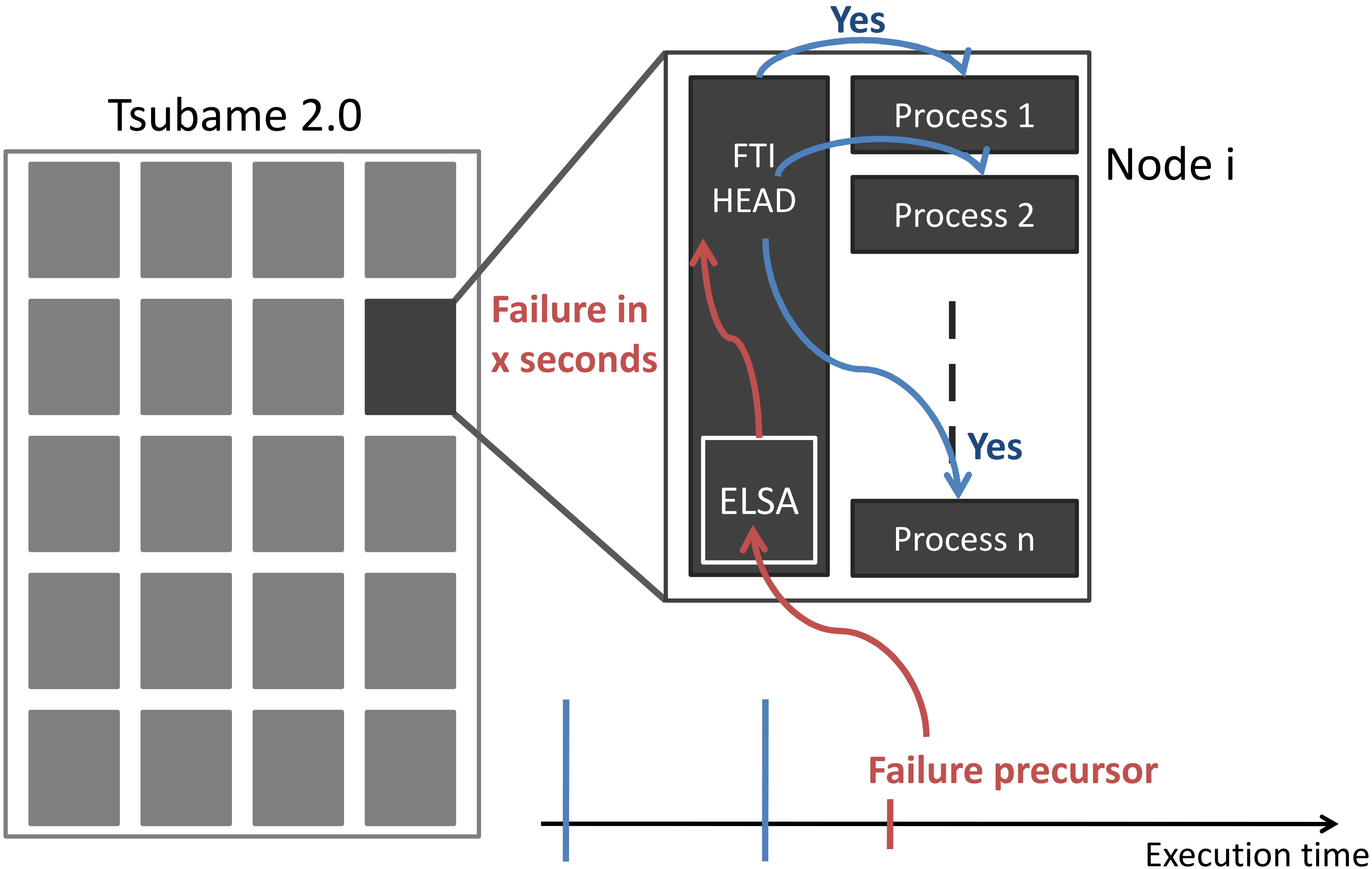
- 2 main contributions:
  - Communication between the Head and application processes
  - **Checking predictions regularly**

- Predictions
  - Every 10s
  - App processes ask every x no of iterations
  - Adapt x dynamically to correspond to 10s

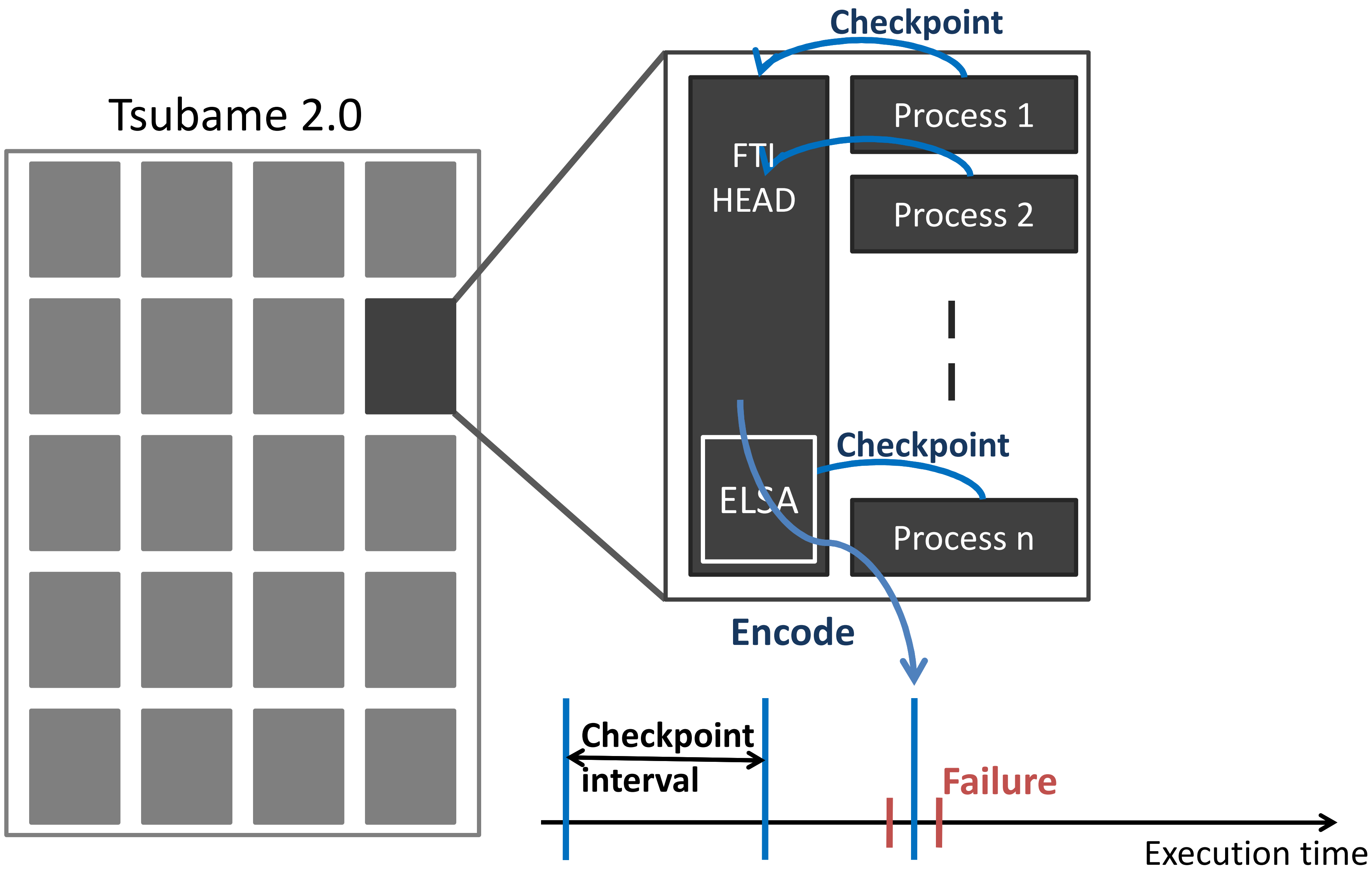
Modifications to FTI



Modifications to FTI



Modifications to FTI



Modifications to ELSA

- As daemon:
  - Monitor the logs at all time
    - Adapt the correlation set
  - Reacts to the incoming stream of events
- Running distributed – application based
  - Looses multi-node errors
  - Save current predictions: Active chains
    - In case the prediction was wrong (FP) – adapts correlations
    - For true positives: positive precursors
  - Reads the log file bottom-up for 10s
    - More general view: Accurate anomaly detection

Modifications to ELSA

- Filtering the prediction send to the Head
  - According to the analytical model
    - Too early cases
  - Predictions that don't leave enough time to take a checkpoint
  - Predictions with low confidence
    - Are added in the suspicious list and are monitored
    - In case a suspicious list is confirmed
      - Adapt true positives cases
  - Predictions with high time lags
    - To decrease the waste – trigger the prediction later

Table of contents

Experiments

- On Tsubame 2.0
- Logs:
  - Tsubame2 logs with synthetic / Tsubame2 correlations
- Different nodes, threads/node
  - 6 executions for each test case – mean
- Overheads include:
  - The preventive and proactive checkpoint waste
  - Protocol specific overheads
    - For example due to the communication between FTI and the application processes
  - Overhead of dedicating 1 extra thread per node for FTI

Test cases

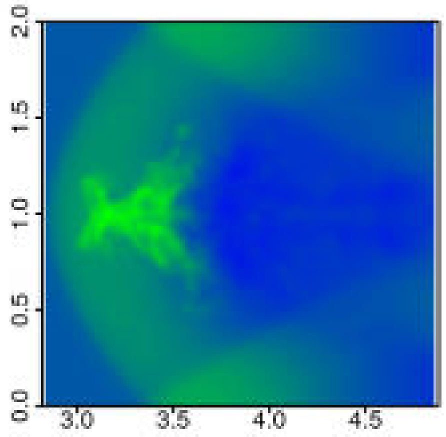
- Failure free execution
  - Measure overheads
    - FTI over no checkpoint
    - ELSA+FTI over initial version of FTI
    - ELSA+FTI over no checkpoint
- False positives
  - Triggers un-necessary checkpoints
  - Measure the overhead

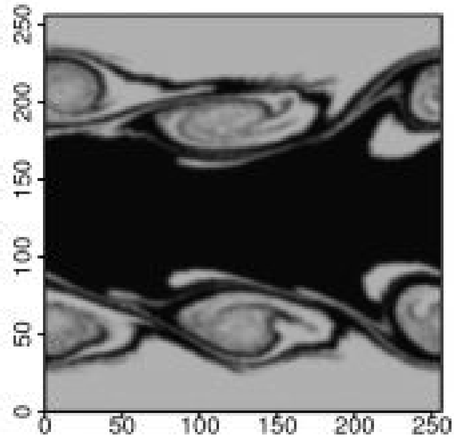
Applications

- Gadget2
  - Code for cosmological N-body simulations on massively parallel computers
  - Uses MPI
  - The same code can be used for
    - studies of isolated systems, for simulations of the cosmological expansion of space
  - Was used for the Millennium Run
    - One of the largest N-body simulation used to investigate how matter in the universe evolved over time
- We use 3 different tests based on Gadget2

Applications

- Test cases
  - Blob test: A spherical cloud of gas is placed in a wind-tunnel with periodic boundary conditions.
    - 100MB checkpoint size
  - Kelvin-Helmholtz test: Two fluids in pressure equilibrium with opposing velocities. The interface between the fluids is perturbed. Records the evolution of mixing the fluids.
    - 100MB checkpoint size





Applications

- Test cases
  - LCM - The galaxy formation process
  - We only ran it for a small number of particles
  - Small example: 10 MB checkpoint size
  - Communication overhead

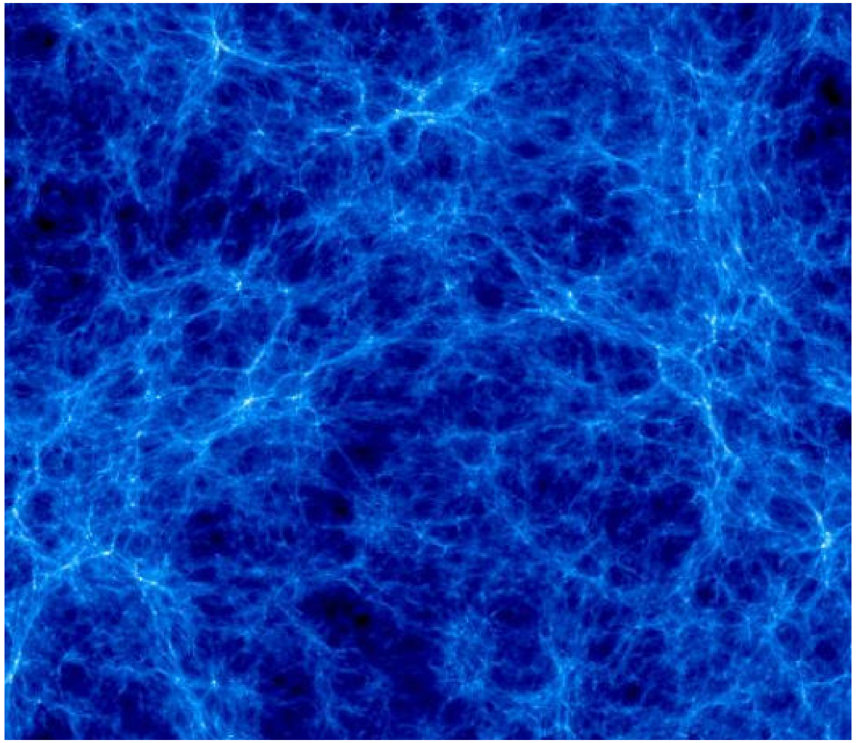
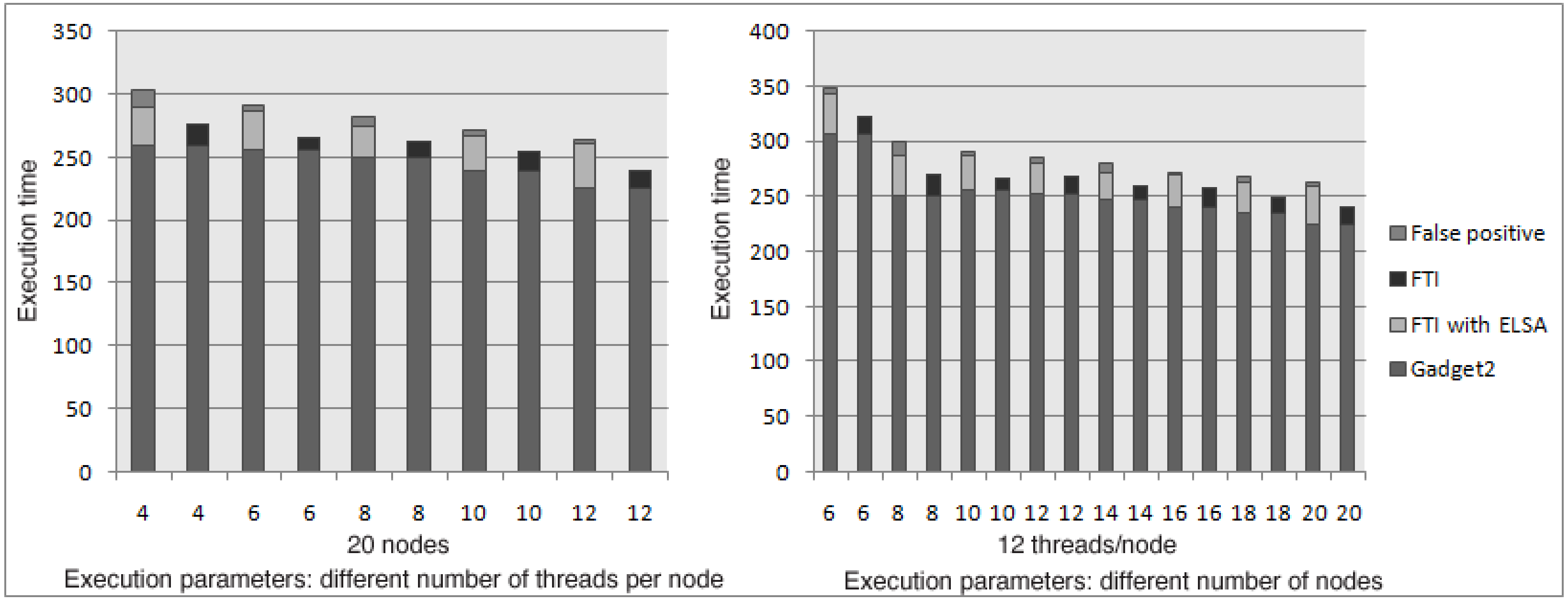


Table of contents

Results

- Changing
  - Number of threads in each node
  - Number of nodes the application executes on



Blob test

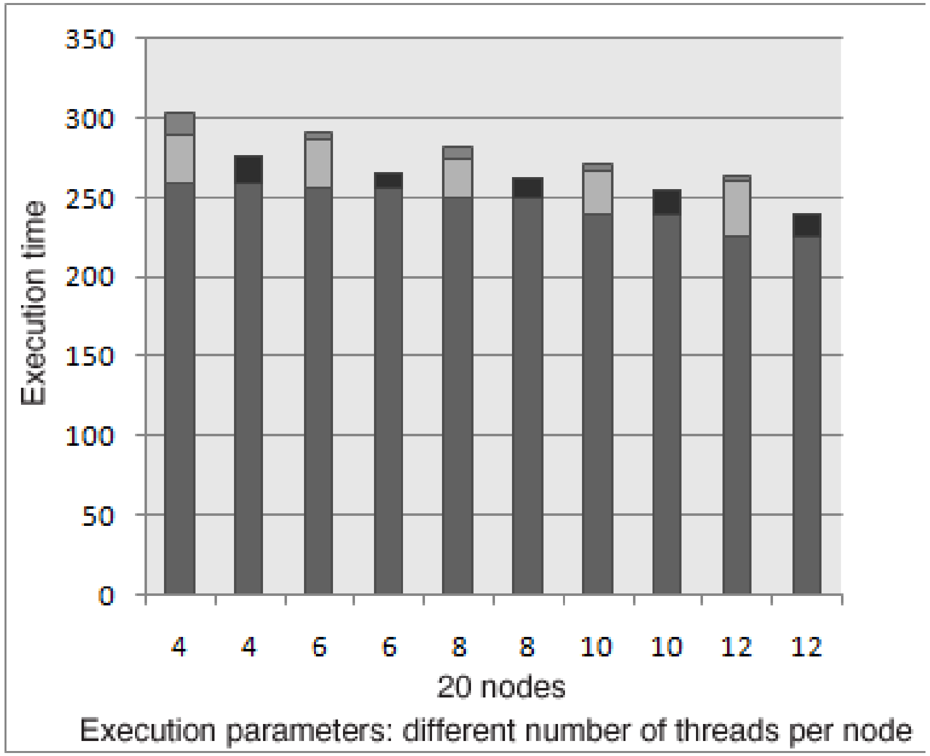
Results

- Changing
  - **Number of threads in each node**
  - Number of nodes the application executes on

Same number of Heads

- Same inter-node communication
- Same checkpoint size per node
- Higher intra-node communication

Constant execution difference



Blob test

Results

- Changing
  - **Number of threads in each node**
  - Number of nodes the application executes on

Same number of Heads

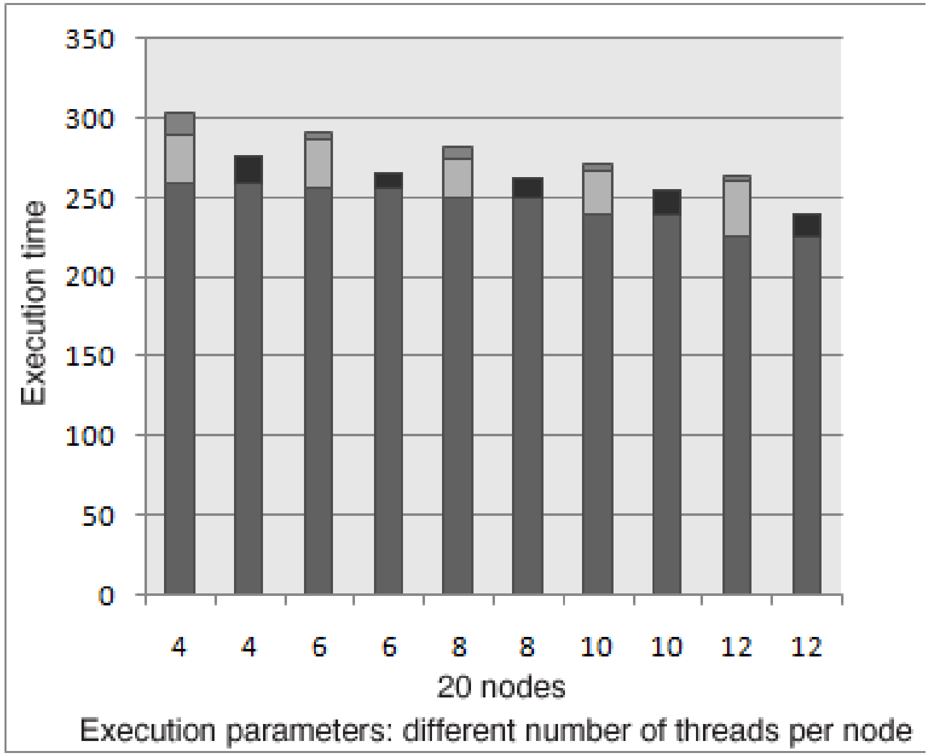
-
-
-

Co

Specific to the Blob test

- Irregular application: iterations have different execution times.
- FTI needs to adapt the check window

No patterns in the overhead



Blob test

Results

- Changing
  - **Number of threads in each node**
  - Number of nodes the application executes on

Same number of Heads

-
-
-

Co

Specific to the Blob test

-

hav

-

wir

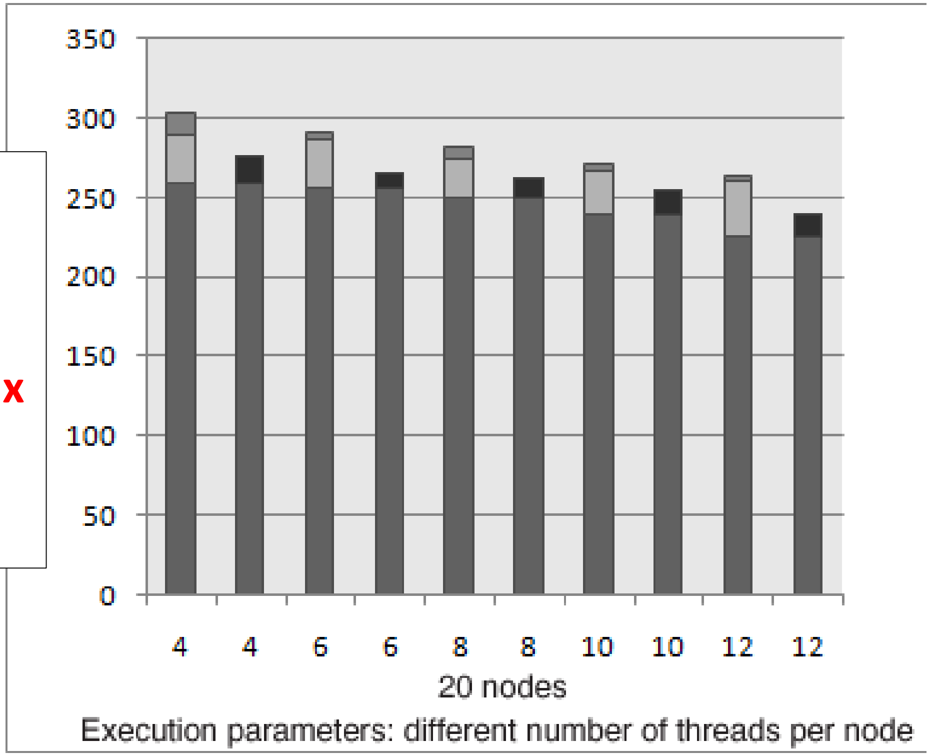
No

Overheads

FTI has around 6%

FTI and ELSA have min 8% and max 11%

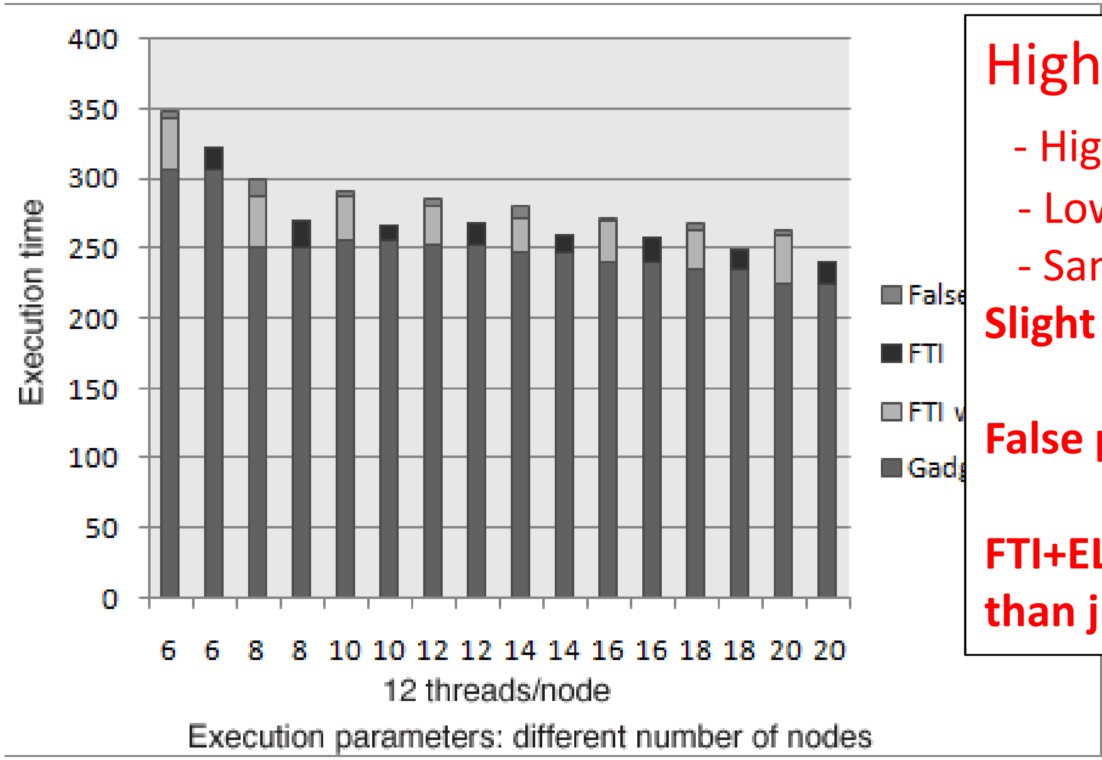
False positives add around 2%



Blob test

Results

- Changing
  - Number of threads in each node
  - **Number of nodes the application executes on**



Higher number of Heads

- Higher inter-node communication
- Lower checkpoint size per node
- Same intra-node communication

Slight increase in overhead

False positives add ~1.5%

FTI+ELSA overhead higher with 2-6% than just FTI

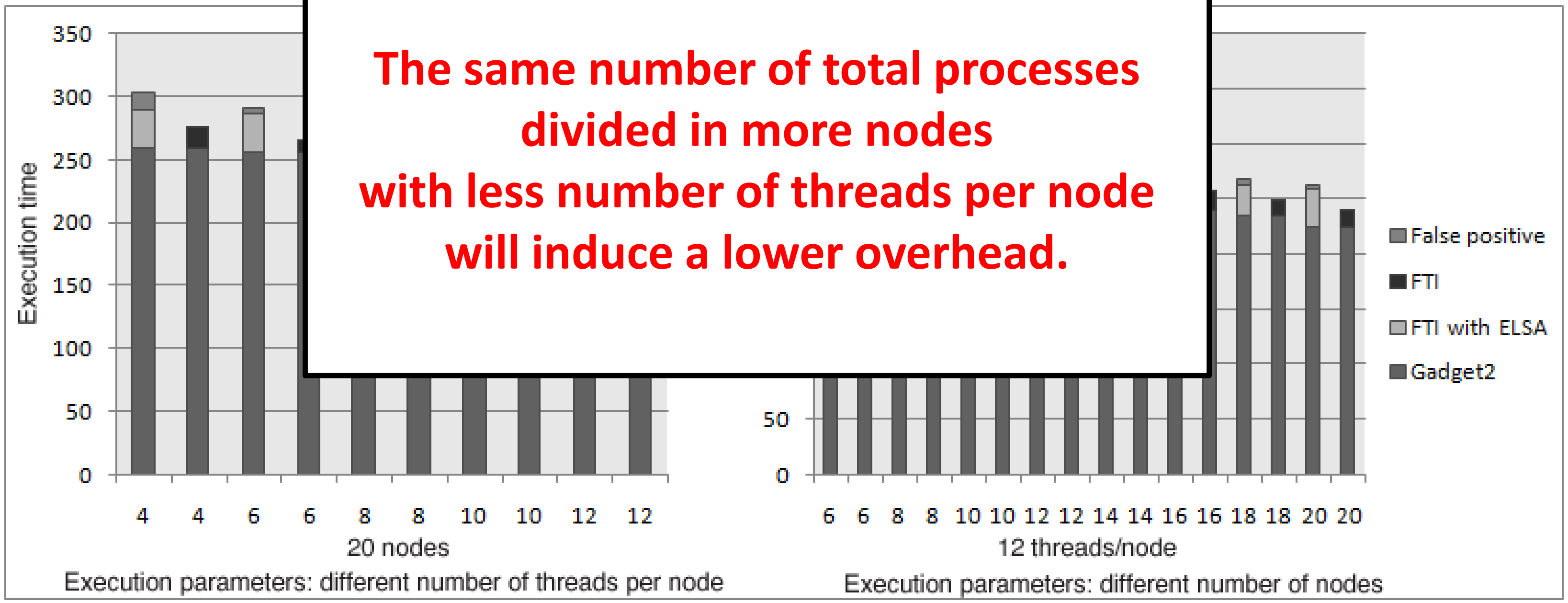
Blob test

Results

- Changing
  - Number of threads in each node
  - Numb tes on

Observation

The same number of total processes divided in more nodes with less number of threads per node will induce a lower overhead.



Blob test

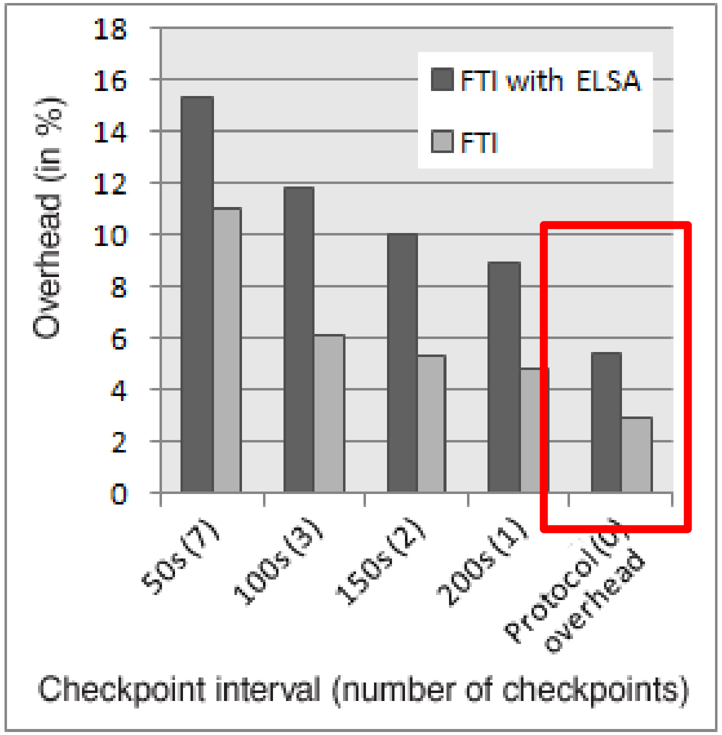
Results

- Overhead for different checkpoint intervals
  - Same number of total processes

Lower bound

- FTI and ELSA running
- No checkpoints are taken
- The overhead is 5.44%

Communication overhead
- Observed on LCM



Blob test

Results

- Different correlation and template set
  - If the analysis of the correlation is <10s
    - No extra overhead
    - For the Tsubame2 correlations the analysis time is ~2s
  - Stress test:
    - 10 times more correlations – 1.3% overhead to ELSA+FTI
- Impact of the checking interval on the number of usable predictions
  - Results for ELSA daemon as baseline
    - Compared with FTI+ELSA with 10s check intervals
    - Recall difference of <1%
    - For check values > 30s the recall value drops
    - Depends on the system

Table of contents

- Possible merging methodologies
- Modifications to FTI and ELSA
- Experiments test cases
- Results
- **Conclusions**
- **Future directions**

Conclusion

- FTI + ELSA
  - Shows ~12% overhead
    - Compared with no-chekpoint
  - With just 2-6% more than FTI alone
    - Mostly because the increase in intra-node communication
  - Baseline of ~5% overhead
  - Small extra overhead due to false positives (~1-2%)
- ELSA
  - Looses multi-node failures – Recall of 40%
  - Looses predictions with small lead time
    - Recall of 32%

Future work

- Fault distribution after prediction
- Include multi-node predictions
  - Without increasing inter-node communication
- Include statistic metrics into the prediction process
  - Precursor detectors for the prediction
  - System degradation prediction
- Predict the un-error periods
  - Decrease the waste due to taking unnecessary checkpoints

Collaboration directions

1) Mathematical models for computing the benefits
   – Collaboration with INRIA / UIUC

2) Combining prediction with other solutions
   2.1) Live migration
   - Collaboration with ANL / INRIA (also IBM)

   2.2) Charm++ fault tolerance
   - Collaboration with UIUC

3) Using ELSA for root cause analysis
   – Collaboration with UIUC / ANL (also Sandia)

4) Understanding failures in HPC: precursor detectors
   – Collaboration with UIUC / ANL

Additional Q&A

Thank you

Ana Gainaru

Coupling failure prediction, proactive and preventive checkpoint for current production HPC systems

Applications

Blob test

Kelvin-Helmholtz test