CIFTS: A Coordinated Infrastructure for Fault Tolerant Systems : *Experiences and Challenges*

Rinku Gupta

Mathematics and Computer Science Division

Argonne National Laboratory

CIFTS Project

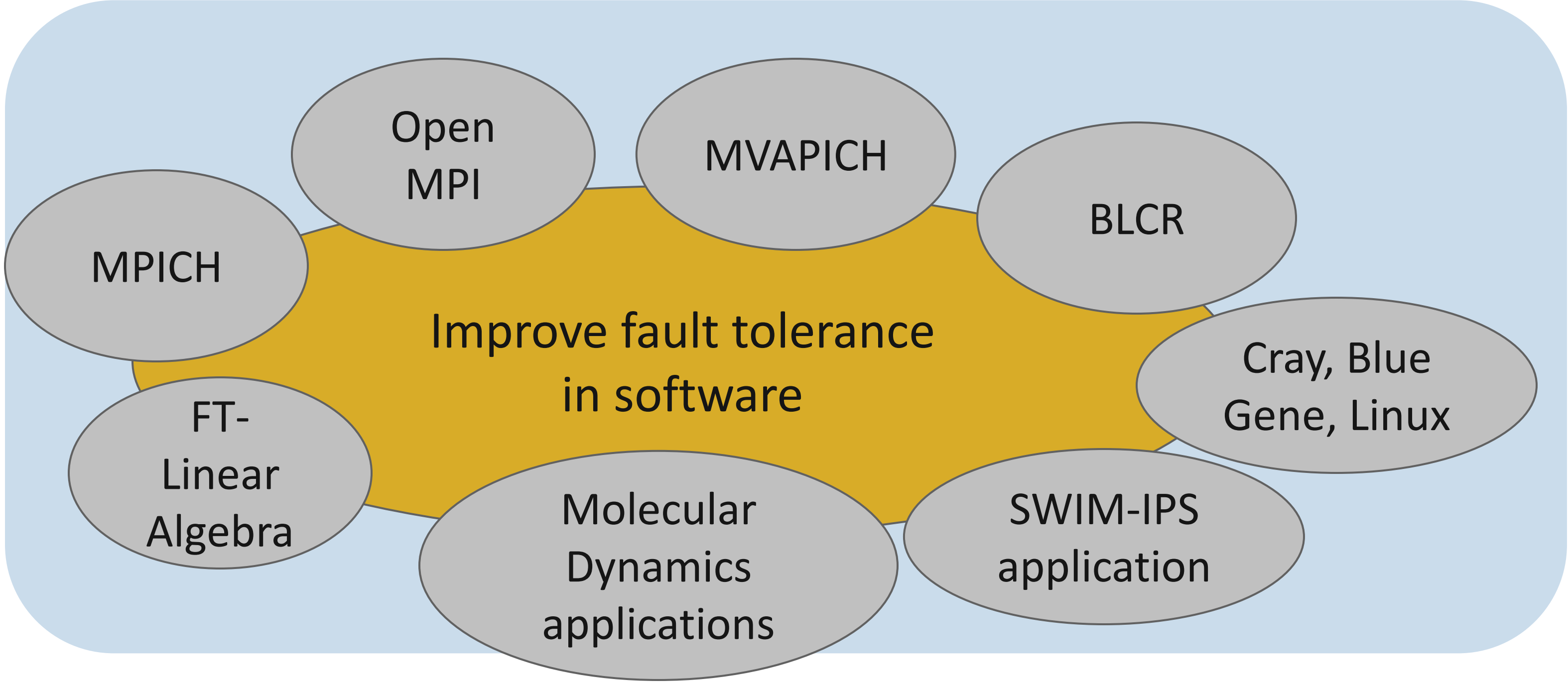
- The CIFTS Project
  - Initiated in 2007
  - Goal: **To Improve End-to-End Fault Tolerance in Systems**

- Team
  - Argonne National Lab : Pete Beckman
  - Oak Ridge National Lab : Al Geist/ David Bernholdt
  - Lawrence Berkeley National Lab: Paul Hargrove
  - University of Tennessee, Knoxville: Jack Dongarra
  - Indiana University: Andrew Lumsdaine
  - Ohio State University: D.K. Panda

Argonne NATIONAL LABORATORY
BERKELEY LAB
INDIANA UNIVERSITY BLOOMINGTON
THE OHIO STATE UNIVERSITY
THE UNIVERSITY of TENNESSEE UT KNOXVILLE
ornl OAK RIDGE NATIONAL LABORATORY

Team

End to End Fault Tolerance



Improve fault tolerance in software

Improving fault tolerance in MPI

- MPICH (ANL)
  - Incorporated system-level checkpoint/restart using BLCR
  - Run-through fault tolerance (process fails, return error and continue)
  - Partial support for MPI 3.1 fault tolerance
- MVAPICH (OSU)
  - Incorporated system-level check pointing/restart using BLCR
  - Pro-active Job migration
  - Automatic InfiniBand path failover
- Open MPI (IU)
  - Transparent, coordinated checkpoint/restart infrastructure
  - Checkpoint-restart enabled process migration
  - Application-level checkpoint Interface
  - Fault Tolerance API using MPI extensions (In development)
- Motivated the push for MPI 3.0 Fault Tolerance standard
- Details: http://www.mcs.anl.gov/research/cifts/publications/index.php

Fault Tolerance in Applications

- Improving fault tolerance in SWIM-IPS, AMBER and LAMPPS application (ORNL)
- For example: AMBER, LAMPPS application
  - Built on hypothesis that certain applications can have "health parameters"
  - Health parameters:
    - Good indicators of overall health of the application progress
    - Can be monitored *(LIVE)* with little or no over-head
    - Deviations from expected behavior can be indication of fault
  - Molecular dynamics (MD): Possible heath parameters : Temperature, Energy (constant energy runs), Simulation volume
  - Manage checkpoint/restart capability based on the health parameters
- FT approach taken varies with application

Improvements in BLCR

- Berkeley Checkpoint/Restart for Linux (LBNL)
  - Single-node checkpointer which cooperates with MPI for distributed applications
- Integrated with MPIs (MPICH, Open MPI and MVAPICH), SLURM and TORQUE
- Several improvements (version 0.9)
  - Coalescing of small I/O requests into larger ones
  - In-kernel compression of checkpoint data
  - Incremental checkpointing and memory-exclusion hints
  - in-place rollback
- Adopted on Cray Systems, and on Blue Gene systems using Linux-derivative kernels (such as ZeptoOS)

"Checkpoint/Restart-Enabled Parallel Debugging"*, J. Hursey and C. January and M. O'Connor and P. Hargrove and D. Lecomber and J. Squyres and A. Lumsdaine, Proceedings of EuroMPI, 2010*

Improving Fault Tolerance in Math libraries

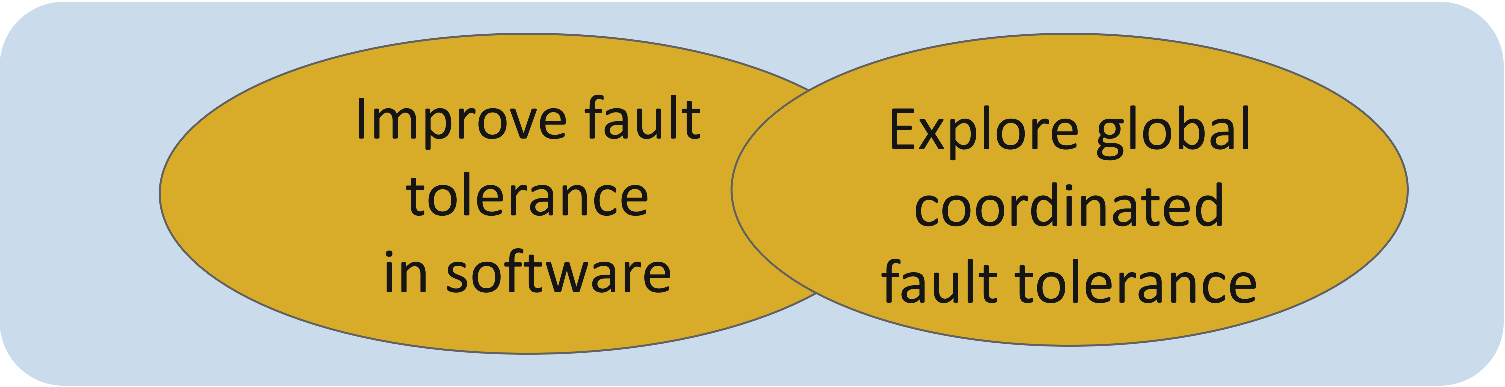
- FT Linear Algebra: Dense linear algebra library (UTK)
- Work done with FT-LA and ScaLAPACK
  - Design on checksum-based fault tolerant algorithms
    - Targeted at BLAS3 kernels such as matrix-matrix multiplication and LU decomposition

“Correlated Set Coordination in Fault Tolerant Message Logging Protocols"*. A. Bouteiller, T. Herault, G. Bosilca and J. Dongarra, Lecture Notes in Computer Science, Proceedings of the 2011 Euro-Par conference, 2011*

“Algorithm-based fault tolerance for dense matrix factorizations”*. Peng Du, Aurelien Bouteiller, George Bosilca, Thomas Herault, and Jack Dongarra.. Technical Report 253, LAPACK Working Note, July 2011.*

“Soft error resilient QR factorization for hybrid system”*, Peng Du, Piotr Luszczek, Stanimire Tomov, and Jack Dongarra, Technical Report 252, LAPACK Working Note, July 2011.*

End to End Fault Tolerance in CIFTS



Improve fault tolerance in software

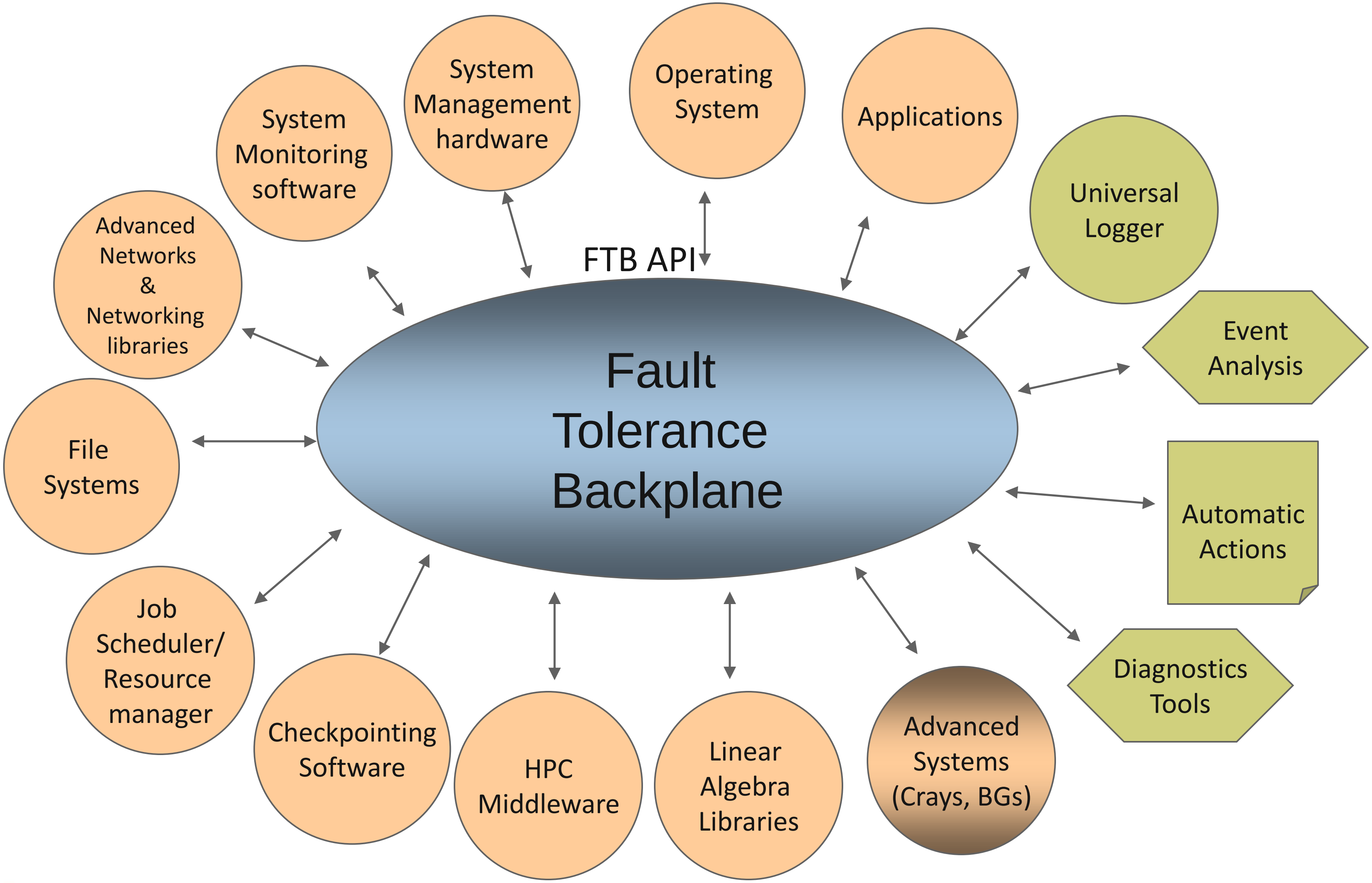
Explore global coordinated fault tolerance

- Open Questions in the last decade
  - Can global fault information improve detection, diagnosis and responsiveness to fault?
  - What are the missing fault-tolerance features in software today?
  - What additional mechanisms, tools, and technologies are needed for coordinated fault tolerance?
  - What standards, outreach, and community interaction are needed for easier adoption?

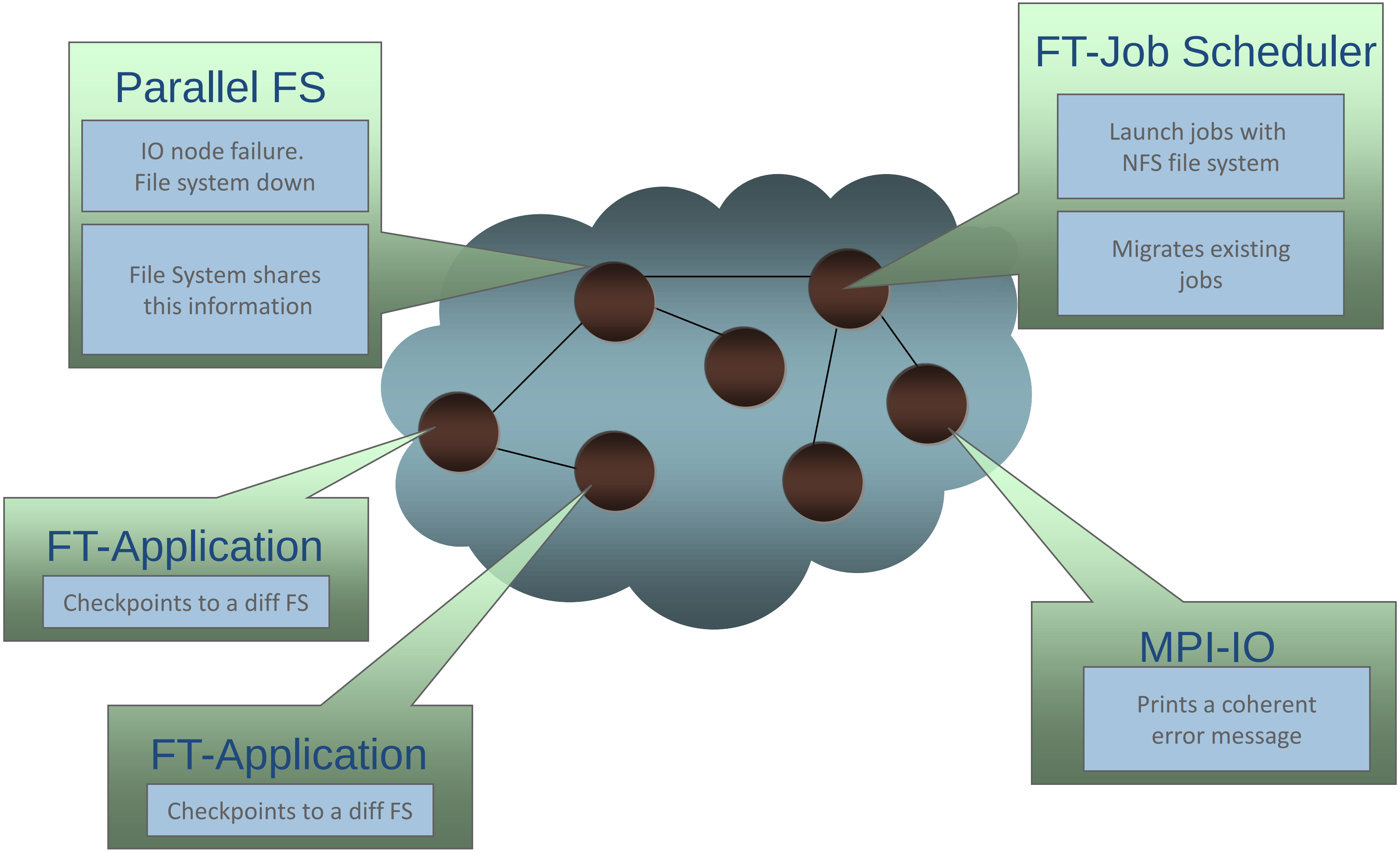
CIFTS approach for coordinated fault tolerance

- CIFTS provides a coordination framework for different components to exchange hardware and software fault information.
  - This communication framework is called ***The Fault Tolerance Backplane***
- Provides a standard FTB API to exchange fault information
- Provide a reference implementation of the FTB API
- Work with a range of widely-used software and plugs them into the CIFTS infrastructure

Fault Tolerance Framework


System Monitoring software
System Management hardware
Operating System
Applications
Universal Logger
Advanced Networks & Networking libraries
FTB API
Fault Tolerance Backplane
Event Analysis
File Systems
Automatic Actions
Job Scheduler/ Resource manager
Checkpointing Software
HPC Middleware
Linear Algebra Libraries
Advanced Systems (Crays, BGs)
Diagnostics Tools

Scenario Using Coordination



Parallel FS

IO node failure.
File system down

File System shares
this information

FT-Job Scheduler

Launch jobs with
NFS file system

Migrates existing
jobs

FT-Application

Checkpoints to a diff FS

FT-Application

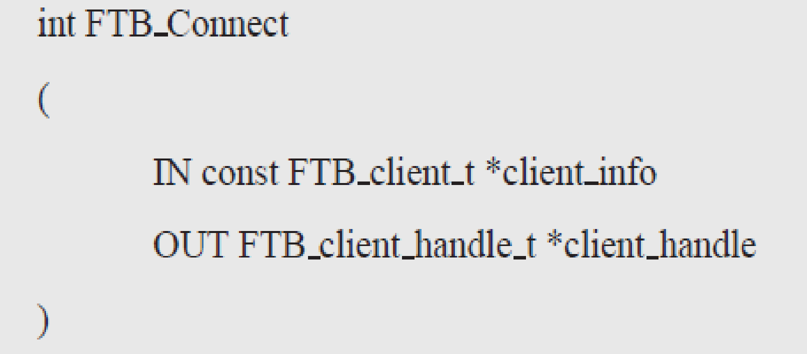
Checkpoints to a diff FS

MPI-IO

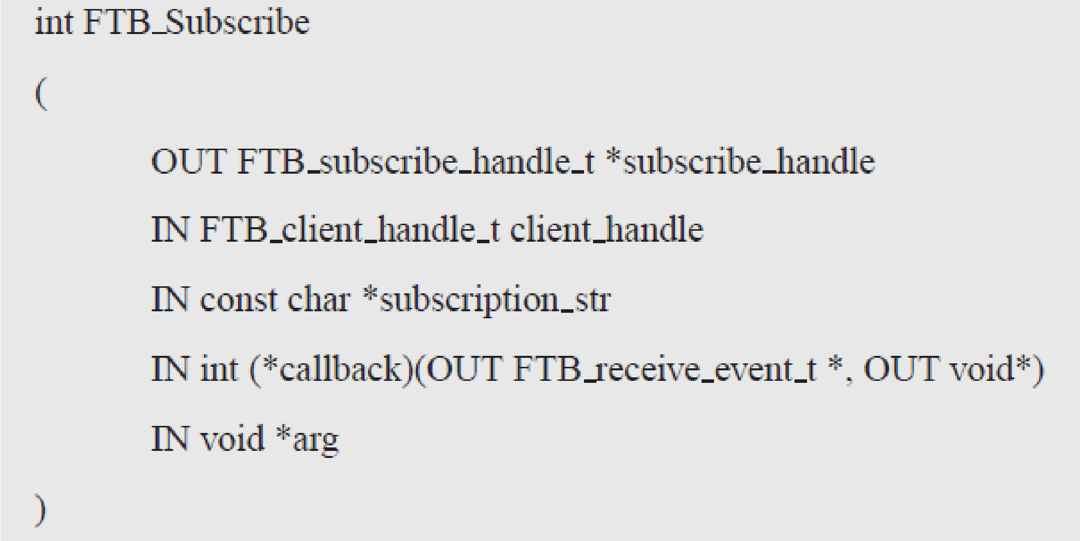
Prints a coherent
error message

The FTB Client API

- Provides a set of simple FTB routines, loosely based on the publish-subscribe principle
  - FTB_Connect()
  - FTB_Publish_event()
    - Events declared in code or via XML files
  - FTB_Subscribe()
    - Various filters
    - Polling vs. asynchronous notification
  - FTB_Poll_for_event()
  - FTB_Unsubscribe()
  - FTB_Disconnect()

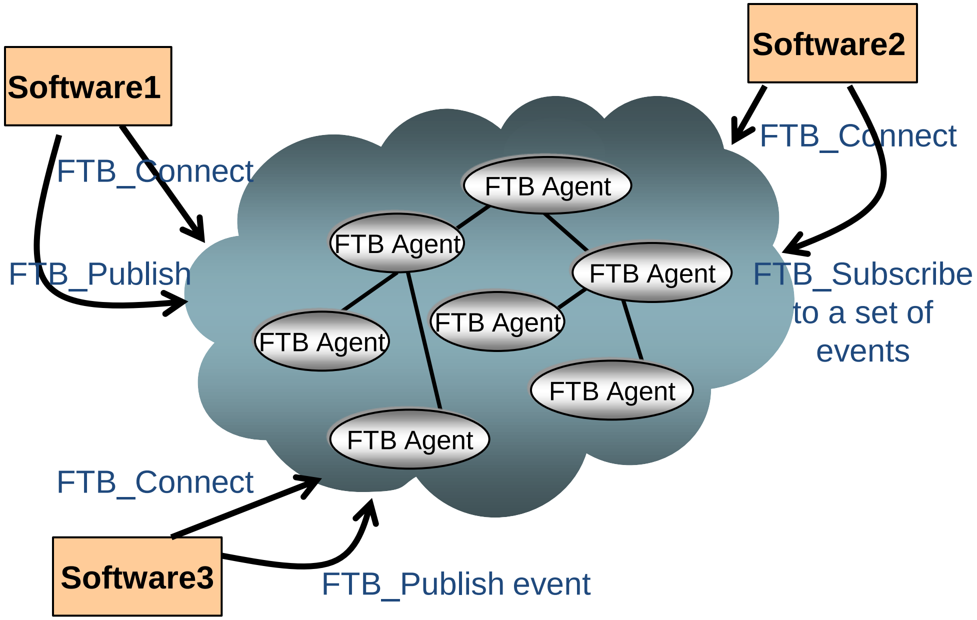


```
int FTB_Connect
(
        IN const FTB_client_t *client_info
        OUT FTB_client_handle_t *client_handle
)
```



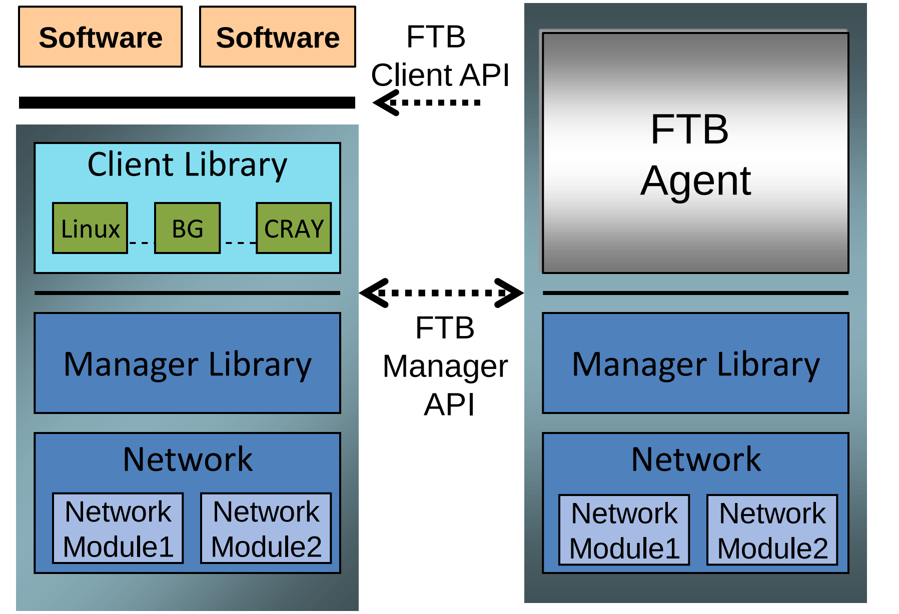
```
int FTB_Subscribe
(
        OUT FTB_subscribe_handle_t *subscribe_handle
        IN FTB_client_handle_t client_handle
        IN const char *subscription_str
        IN int (*callback)(OUT FTB_receive_event_t *, OUT void*)
        IN void *arg
)
```

The FTB Software Architecture



- The FTB software is a reference implementation of the FTB API

- It is a distributed, self-healing and a highly-scalable framework, capable of handling large number of events and dealing with event storms

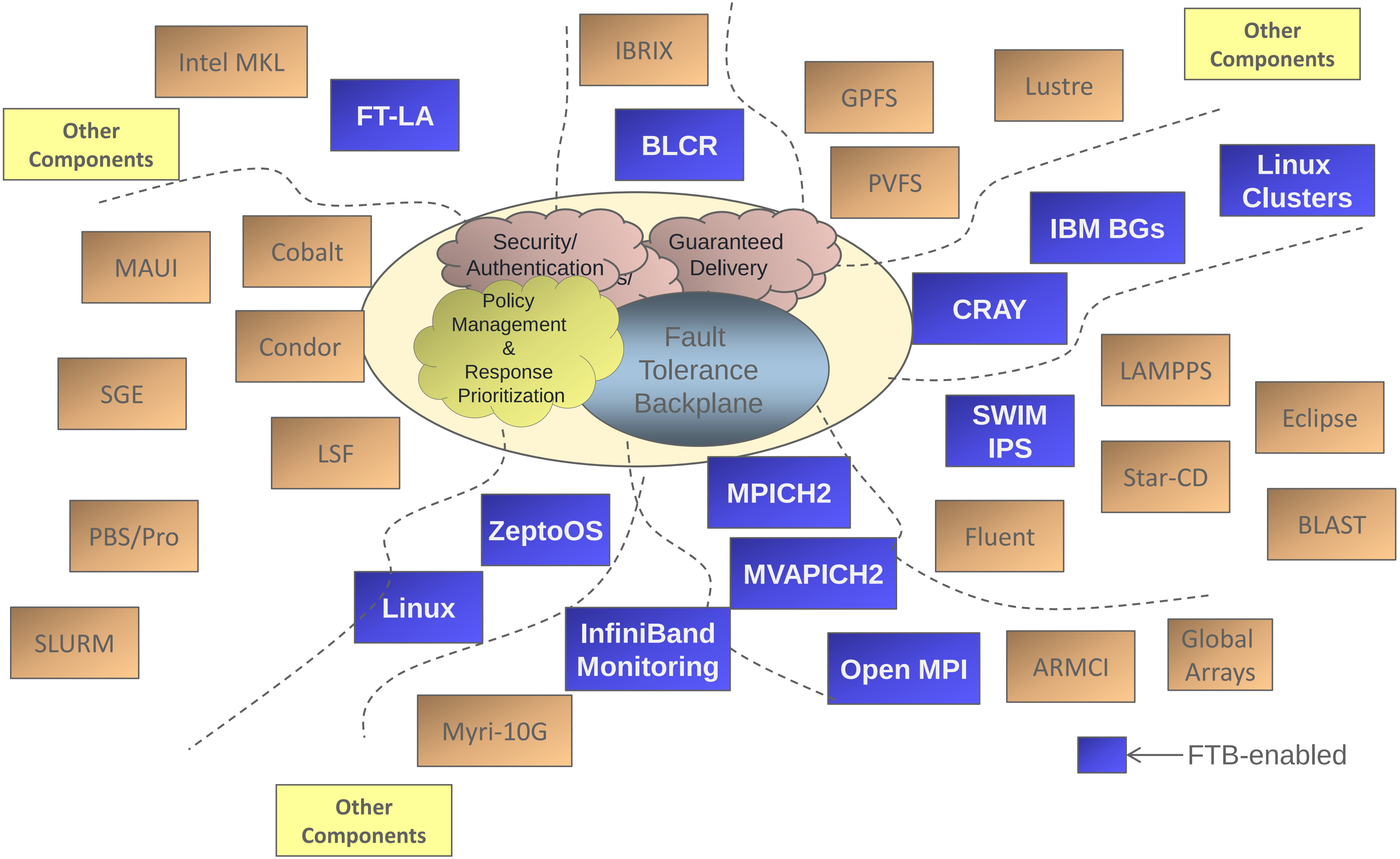


Component Software stack

FTB Agent Software stack

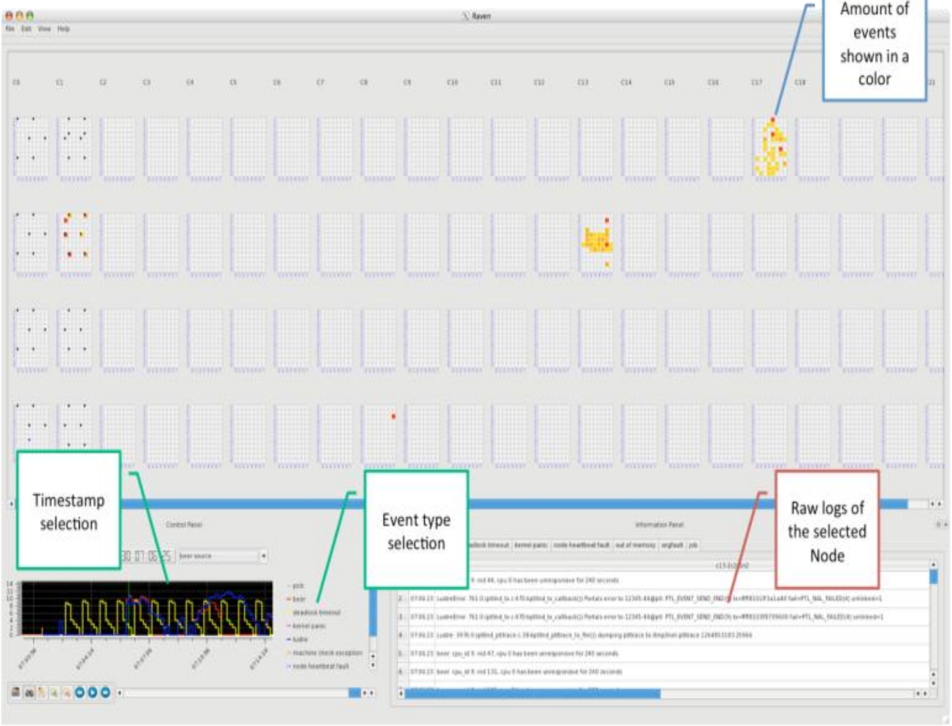
- NOTE: In addition to this implementation, we have another proof-of-concept implementation of FTB API with AMQP (using Apache QPID)

Current State of FTB-enabled software



RAVEN: RAS Data Analysis Through Visually Enhanced Navigation

- FTB-enabled RAS component bridges CRAY RAS event stream to FTB
- User explores event correlations on a physical system map
- Use detailed offline RAS log database to aid fault analysis

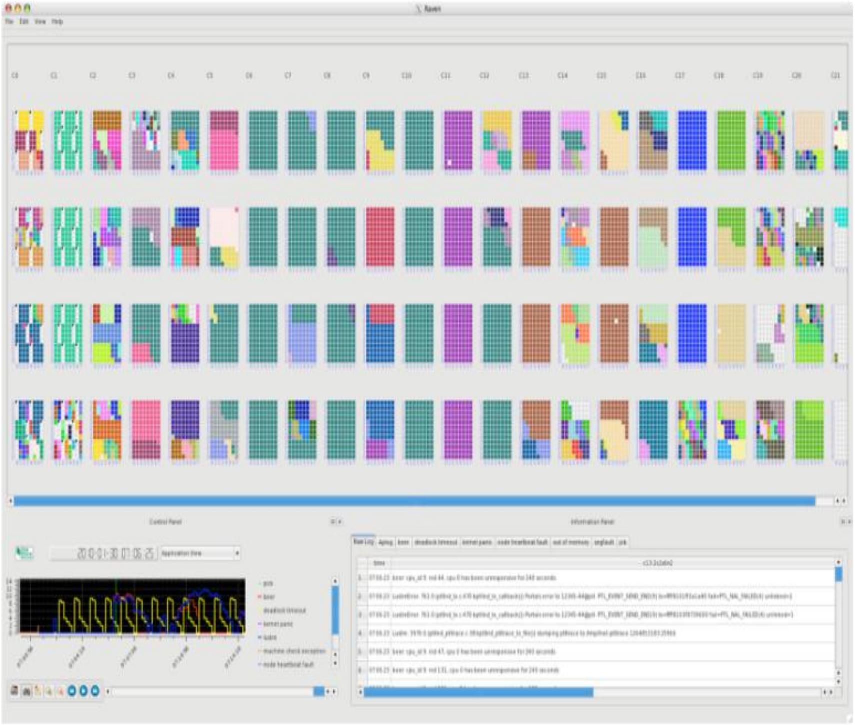


Amount of events shown in a color

Timestamp selection

Event type selection

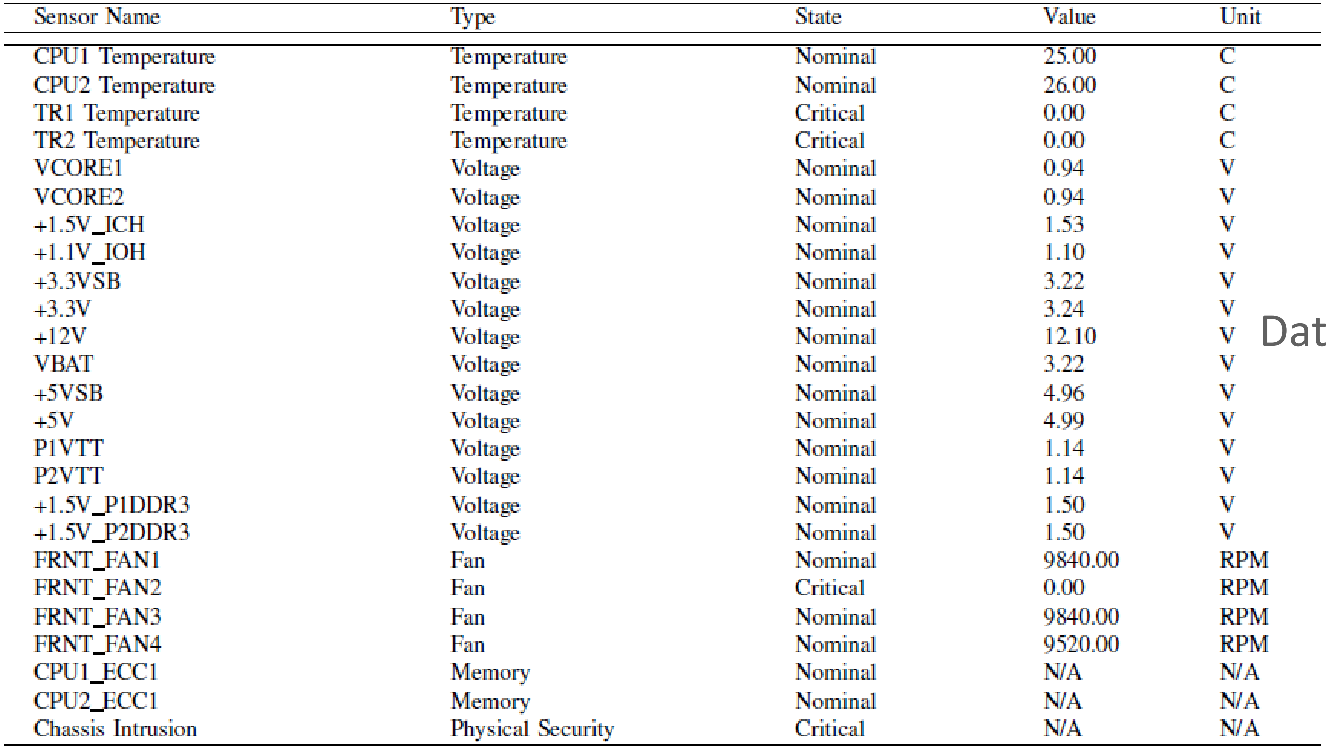
Raw logs of the selected Node



*"**Dynamic Meta-Learning for Failure Prediction in Large-scale Systems: A Case Study**", J. Gu, Z. Zheng, Z. Land, J. White, E. Hocks and B-H Park, Proceedings of the International Conference on Parallel Processing (ICPP), 2008.*

FTB-IPMI (1)

- Intelligent Platform Management Interface (IPMI) defines a set of common interfaces to a computer system which can be used to monitor system health
- FTB-IMPI software is based on FreeIPMI, based on IPMI v1.5/2.0 specification



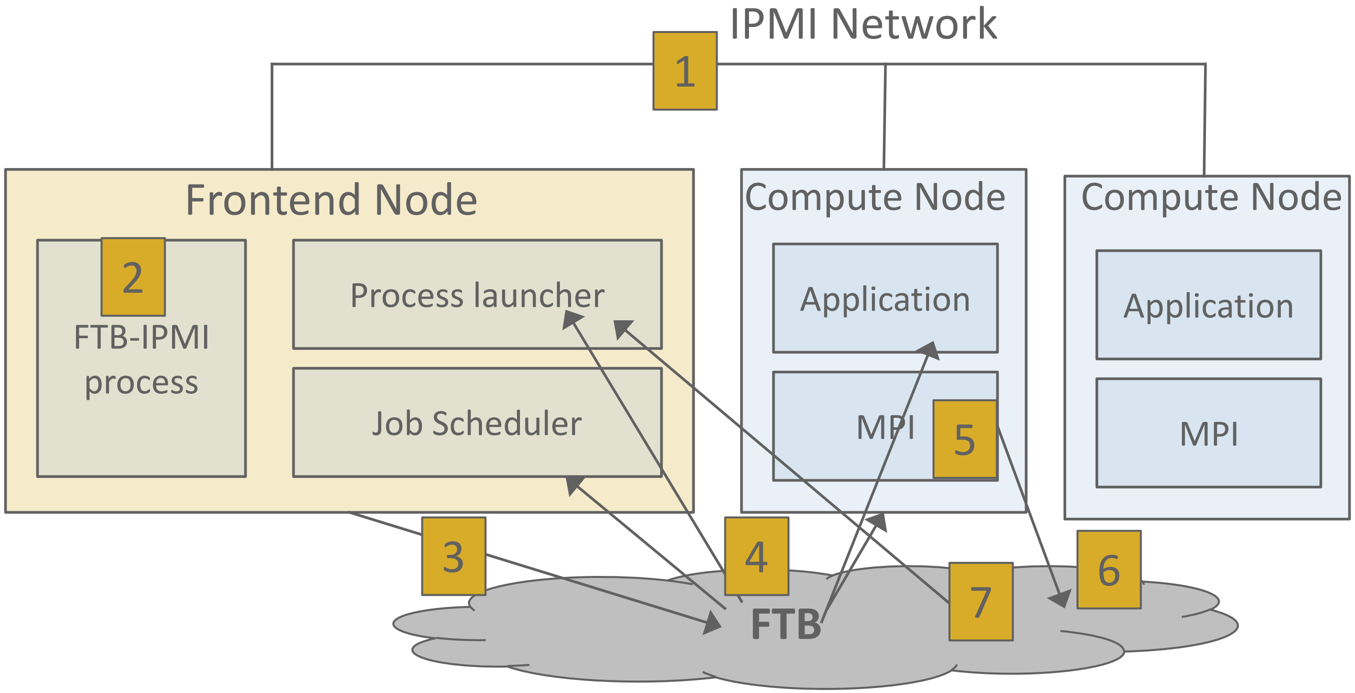
Sensor Name	Type	State	Value	Unit
CPU1 Temperature	Temperature	Nominal	25.00	C
CPU2 Temperature	Temperature	Nominal	26.00	C
TR1 Temperature	Temperature	Critical	0.00	C
TR2 Temperature	Temperature	Critical	0.00	C
VCORE1	Voltage	Nominal	0.94	V
VCORE2	Voltage	Nominal	0.94	V
+1.5V_ICH	Voltage	Nominal	1.53	V
+1.1V_IOH	Voltage	Nominal	1.10	V
+3.3VSB	Voltage	Nominal	3.22	V
+3.3V	Voltage	Nominal	3.24	V
+12V	Voltage	Nominal	12.10	V
VBAT	Voltage	Nominal	3.22	V
+5VSB	Voltage	Nominal	4.96	V
+5V	Voltage	Nominal	4.99	V
P1VTT	Voltage	Nominal	1.14	V
P2VTT	Voltage	Nominal	1.14	V
+1.5V_P1DDR3	Voltage	Nominal	1.50	V
+1.5V_P2DDR3	Voltage	Nominal	1.50	V
FRNT_FAN1	Fan	Nominal	9840.00	RPM
FRNT_FAN2	Fan	Critical	0.00	RPM
FRNT_FAN3	Fan	Nominal	9840.00	RPM
FRNT_FAN4	Fan	Nominal	9520.00	RPM
CPU1_ECC1	Memory	Nominal	N/A	N/A
CPU2_ECC1	Memory	Nominal	N/A	N/A
Chassis Intrusion	Physical Security	Critical	N/A	N/A

IPMI
Data from a single node

FTB IPMI (2)

2

IPMI Sensor Event		FTB Action
State change to *Nominal*	⇒	Publish event (Severity INFO)
State change to *Warning*	⇒	Publish event (Severity WARNING)
State change to *Critical*	⇒	Publish event (Severity WARNING)
Read Error	⇒	Publish event (Severity CRITICAL)

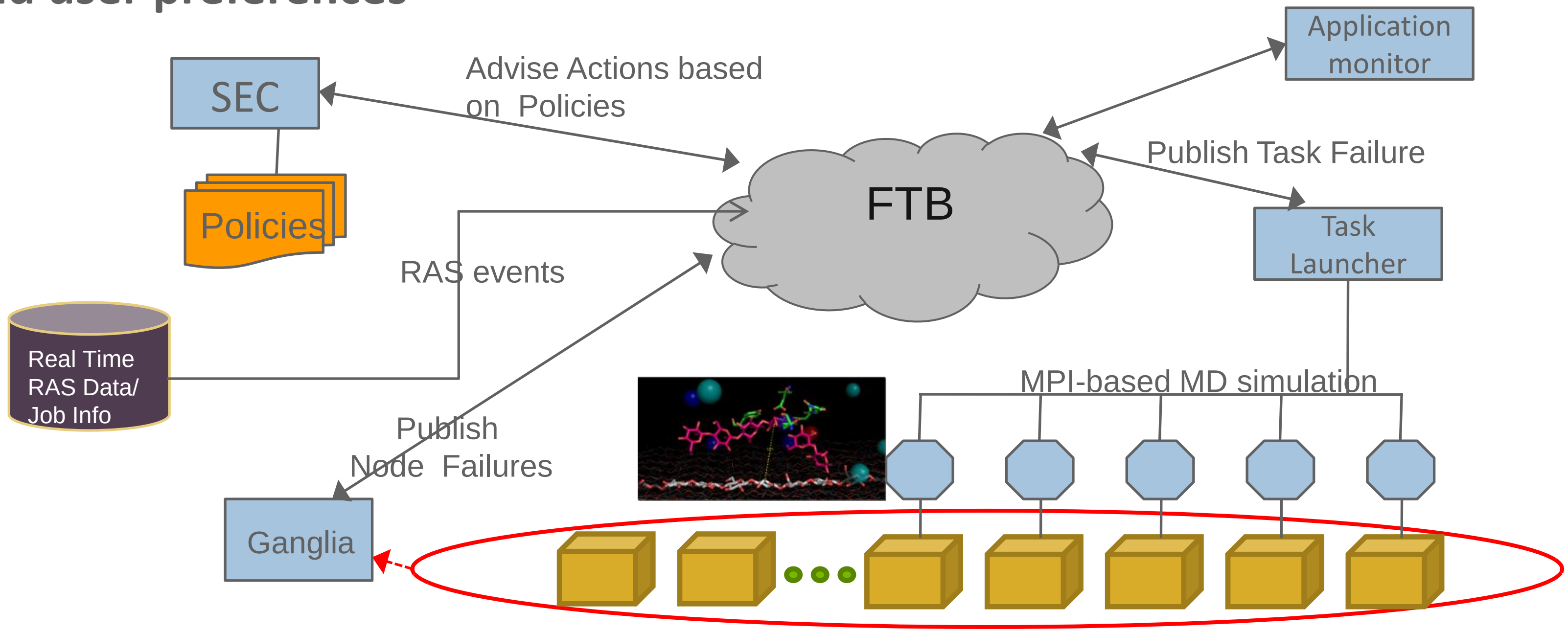


1. Gather sensor data on IPMI network
2. FTB-IPMI applies rules and
3. Publishes FTB event
4. Deliver FTB event to subscribers
5. MVAPICH uses prediction engine and
6. Publishes predicted event (like node_failure)
7. Launcher carries out migration

*** "Monitoring and Predicting Hardware Failures in HPC Clusters with FTB-IPMI"**, *R. Rajachandrasekar, X. Besseron and D. K. Panda, Proceedings of the International Workshop on System Management Techniques, Processes, and Services (SMTPS), in conjunction with International Parallel and Distributed Processing Symposium (IPDPS), 2012*

Application Example: End-to-End Application Fault Response with Molecular Dynamic application

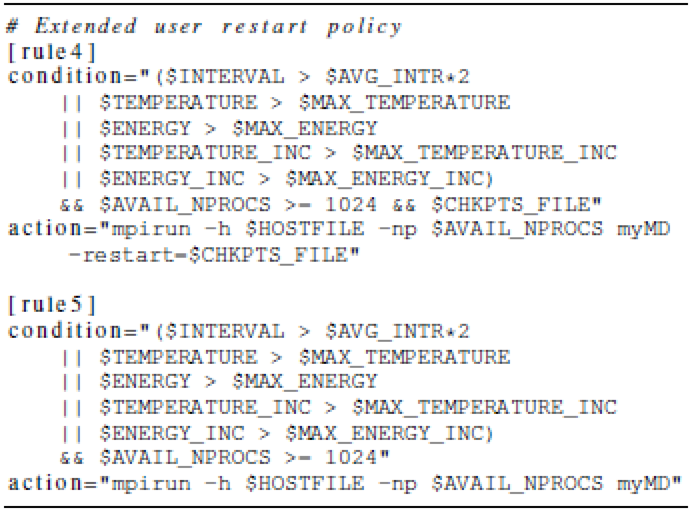
D**etermine best restart option for failed user application based on system state and user preferences**



***"Realization of User-Level Fault Tolerance Policy Management through a Holistic Approach for Fault Correlation"**, B-H. Park, T. Naughton, P. Agarwal, D. Bernholdt, A. Geist and J. Tippens, IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY), June 2011*

***"Application Self-health Monitoring for Extreme-scale Resiliency using Cooperative Fault Management"**, Pratul Agarwal, Thomas Naughton, S. Alam, B-H Park, David Bernholdt, Josh Hursey and Al. Geist, Concurrency and Computation: Practice and Experience (2012). Submitted.*

Application Example: End-to-End Application Fault Response with Molecular Dynamic application



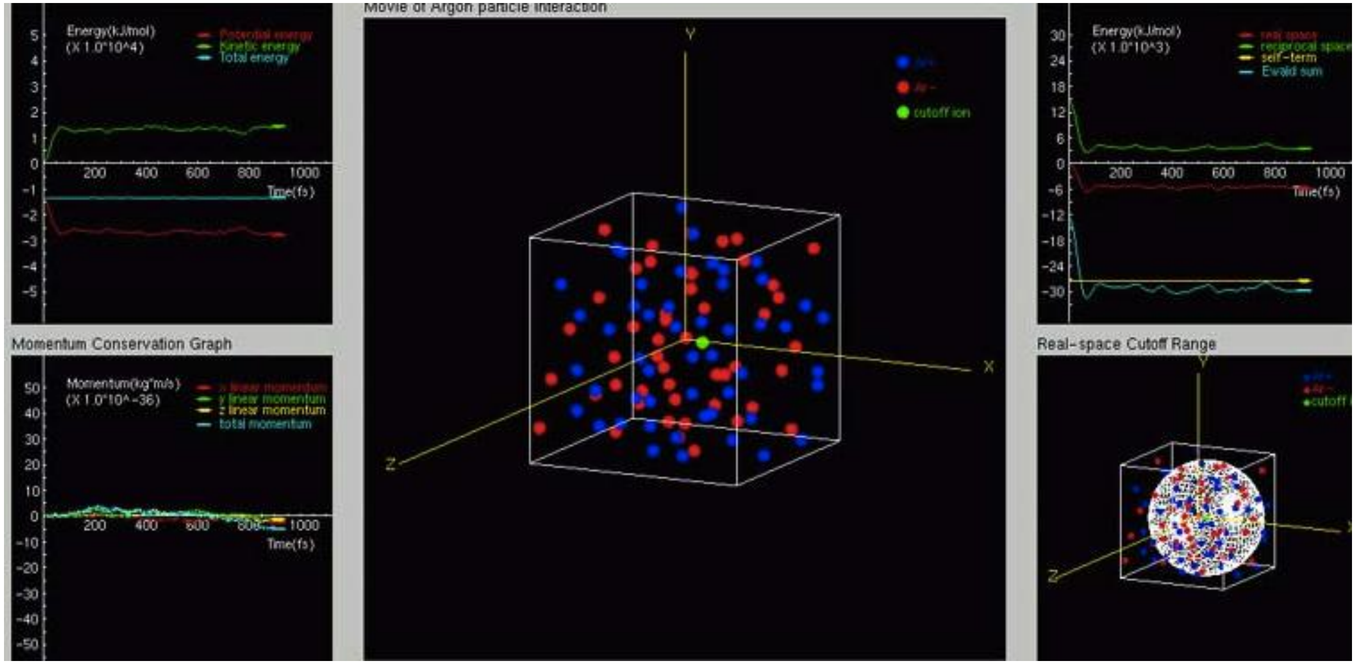
```
# Extended user restart policy
[rule4]
condition="($INTERVAL > $AVG_INTR*2
    || $TEMPERATURE > $MAX_TEMPERATURE
    || $ENERGY > $MAX_ENERGY
    || $TEMPERATURE_INC > $MAX_TEMPERATURE_INC
    || $ENERGY_INC > $MAX_ENERGY_INC)
    && $AVAIL_NPROCS >= 1024 && $CHKPTS_FILE"
action="mpirun -h $HOSTFILE -np $AVAIL_NPROCS myMD
     -restart=$CHKPTS_FILE"

[rule5]
condition="($INTERVAL > $AVG_INTR*2
    || $TEMPERATURE > $MAX_TEMPERATURE
    || $ENERGY > $MAX_ENERGY
    || $TEMPERATURE_INC > $MAX_TEMPERATURE_INC
    || $ENERGY_INC > $MAX_ENERGY_INC)
    && $AVAIL_NPROCS >= 1024"
action="mpirun -h $HOSTFILE -np $AVAIL_NPROCS myMD"
```

The MD application user policy

- Integrity of MD application can be measured by:
  - Wall clock time each simulation step
    - For ex: if wall clock time > 2*statistical average → suspicion that something is wrong
  - MD outputs
    - Temperature, energy, velocity
- Data is sent via application monitor

Application Example: Proactive Fault tolerance with Molecular Dynamic application

- MD application: proxy-code simulating Argon particles interaction
  - Fault predictor to predict faults (via failure trends on the FTB)
  - Transparent process migration using MPI in event of a node failure
  - Simplified Video: http://www.mcs.anl.gov/research/cifts/talks/index.php



Overview of some more FTB-enabled tools (1)

- *FTB-enabled RAS Monitoring tool*
  - Software that polls on RAS database on service node, converts admin-specified RAS events to FTB events and publishes them to FTB
- *FTB-enabled Blue Gene Administrative tool*
  - Gets RAS information from the FTB-enabled RAS monitoring tool
  - Carries out administrative-directed action (email, diagnosis)
- *FTB-enabled Failure Prediction tool*
  - Gets RAS information from the FTB-enabled RAS monitoring tool
  - Uses our failure-prediction research on Blue Gene/P to predict failures (RAS events, job logs) *
  - Prediction : lead time + location of failures**
  - Uses FTB to publish this information

** **"Co-Analysis of RAS Log and Job Log on Blue Gene/P"**, Z. Zheng, L. Yu, W. Tang, Z. Land, R. Gupta, N. Desai, S. Coghlan, and D. Buettner, 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS' 11), May 2011*

** **"System log Pre-processing to Improve Failure Prediction"**, Z. Zheng, Z. Land, B-H Park, and A. Geist, Proceedings of the 39th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2009.*

*** **"A Practical Failure Prediction with Location and Lead Time for Blue Gene/P"**, Z. Zheng, Z. Land, R. Gupta, S. Coghlan, and P. Beckman, Proceedings of the 1st Workshop on Fault-Tolerance for HPC at Extreme Scale (FTXS), in conjunction with DSN'10, 2010*

Overview of some FTB-enabled tools (2)

- FTB Syslog
  - Converts syslog messages into FTB events
- FTB-enabled generic tools
  - FTB Watchdog
  - FTB Publisher,
  - FTB Subscriber,
  - FTB Pingpong,
  - FTB All-to-All
  - FTB Loggers (synchronous, asynchronous)
- FTB-InfiniBand Monitoring tool
  - Used for monitoring InfiniBand network
  - Integrated with MVAPICH (MPI can react based on monitored information)
- FTB-IPMI
  - Tool publishes IPMI information to FTB

** **"Monitoring and Predicting Hardware Failures in HPC Clusters with FTB-IPMI"**, R. Rajachandrasekar, X. Besseron and D. K. Panda, Proceedings of the International Workshop on System Management Techniques, Processes, and Services (SMTPS), in conjunction with International Parallel and Distributed Processing Symposium (IPDPS), 2012*

Accomplishments (Software, Specs and Tools)

- Fault Tolerance Backplane (FTB) API specification (version 0.5)
- FTB software (latest version 0.6)
  - For IBM BG/L and BG/P (ZeptoOS), CRAY and Linux machines
- FTB MPI standardized events (MPICH, MVAPICH, Open MPI)
- FTB-integrated software : MPICH, MVAPICH, Open MPI, BLCR, FT-LA, SWIM-IPS/MD applications
- FTB-enabled tools and libraries for fault logging, fault monitoring, fault analysis and prediction for CRAY, Blue Gene and Linux systems
  - FTB-InfiniBand for monitoring, FTB-syslog, RAVEN (monitoring for CRAY systems) for logging and publishing faults
- For downloads, publications, demos and more information: www.mcs.anl.gov/research/cifts

Fault Coordination frameworks: Challenges (Future Research)

Event Storms

- Single symptom storms
  - Relatively easy, if emerging from a single source
  - Identify duplicate events based on source and event attributes in a time interval
  - Throttle events at the source
  - Supported in FTB
- Different symptom storms
  - Different events from different sources
  - Very tough; require root cause analysis (big research area)
  - Can reduce gravity by throttling of single symptom events

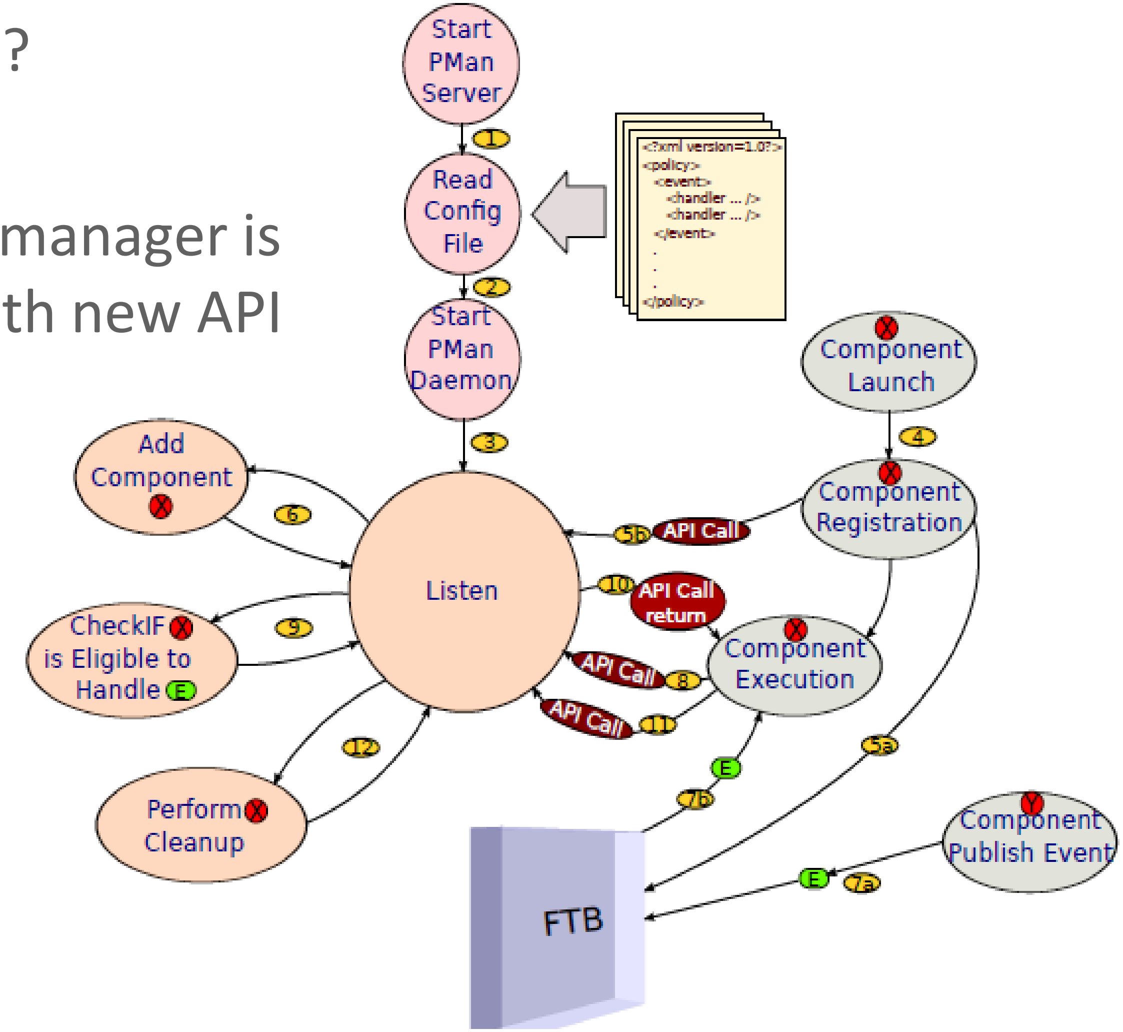
FTB Event Standardization Effort

- FTB-enabled software can publish any fault events they wish
  - Need to have well-defined semantics
  - Many expected to be package/software-specific
  - Even better to standardize across many packages within a category → sister standards
- Standardization of events is a community-wide effort
  - ftb.* portion of namespace reserved for standardized events
  - MPI events have been standardized for MPICH, MVAPICH, and Open MPI teams
  - However, events need to be standardized across other domains (job schedulers: relatively easy, applications: difficult)

Policy Management and Response Negotiation

- Who can take an action for a received event and in what priority?

- Current approach : Policy manager is independent from FTB, with new API
- Current prototype is limited
  - Global policy (specify priority of every component for an event)
  - Increased latency
- Starting point for defining scenarios that work



Policy Management and Response Negotiation

- Policy management in challenging
  - Who determines the **response priority**?
    - Administrator (global view) vs. user (local view)
    - Is this going to be job-based and user-based?
    - Is this going to be process stack-based?
- Software dynamically joins and leaves the system. You cannot predict which software might join and what it might throw
- Software that has priority is responding might exit before responding
- Needs to be reliable, distributed and scalable

Group Aggregators and Query Interfaces

- Independent software that aggregate information and analyze system-wide information
- Why are they needed?
  - Job Scope: Get events published by my job
  - Service Scope: Get all nodes which are running PVFS daemons
  - Get all jobs that are running on a node and determine their job id
  - What all software are FTB-enabled and running on the system?
  - Who can **react** to event Foo ?

Bridging the semantic gap

- Semantic gaps exists between different layers of a software
- Scenario:
  - Network publishes “network communication error with IB adapter = X”
  - Application expects “node hostname has failed ”

Security / Authentication

- Can be a major concern for big production environments
- Sharing fault information with users is not always a good idea
  - With naïve users: Increased support calls (app killed, FTB reported system error, user wants refund of reservation time)
  - With savvy users: how do you deal with policies and still give application control?
- Solutions need research
  - Authentication and security needed
  - Tie authentication levels to “fault events”? to users? Who does this? Is it practical?
  - Limiting consumers to receive only “events within the same job” is not sufficient. Consumers exist beyond jobs!

CIFTS Open Community Model

For more information:

- Web Page: http://www.mcs.anl.gov/research/cifts
- Open SVN Repository: https://svn.mcs.anl.gov/repos/cifts
- Wiki: http://wiki.mcs.anl.gov/cifts/index.php
- TRAC: http://trac.mcs.anl.gov/projects/cifts/wiki
- Mailing lists: cifts_discuss@googlegroups.com

Backup

Go to ”Insert (View) | Header and Footer" to add your organization, sponsor, meeting name here; then, click "Apply to All"