



Argonne
NATIONAL
LABORATORY

... for a brighter future



U.S. Department
of Energy

UChicago ►
Argonne_{LLC}



**Office of
Science**

U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC

Faults and Fault-Tolerance on the Argonne BG/P System

Rinku Gupta

Argonne National Laboratory

Agenda

- ❖ Evolution and Overview of the ANL BG/P System
- ❖ Understanding FT capabilities and challenges of BG/P system
- ❖ Observed Fault trends for the BG/P
- ❖ CIFTS: A new coordinated approach for Fault Tolerant systems

Disclaimer : Data in this presentation has not been quantified, nor mined for differentiation between real test runs and system stress tests. Some of this information is based on best guesses from administrators and application teams

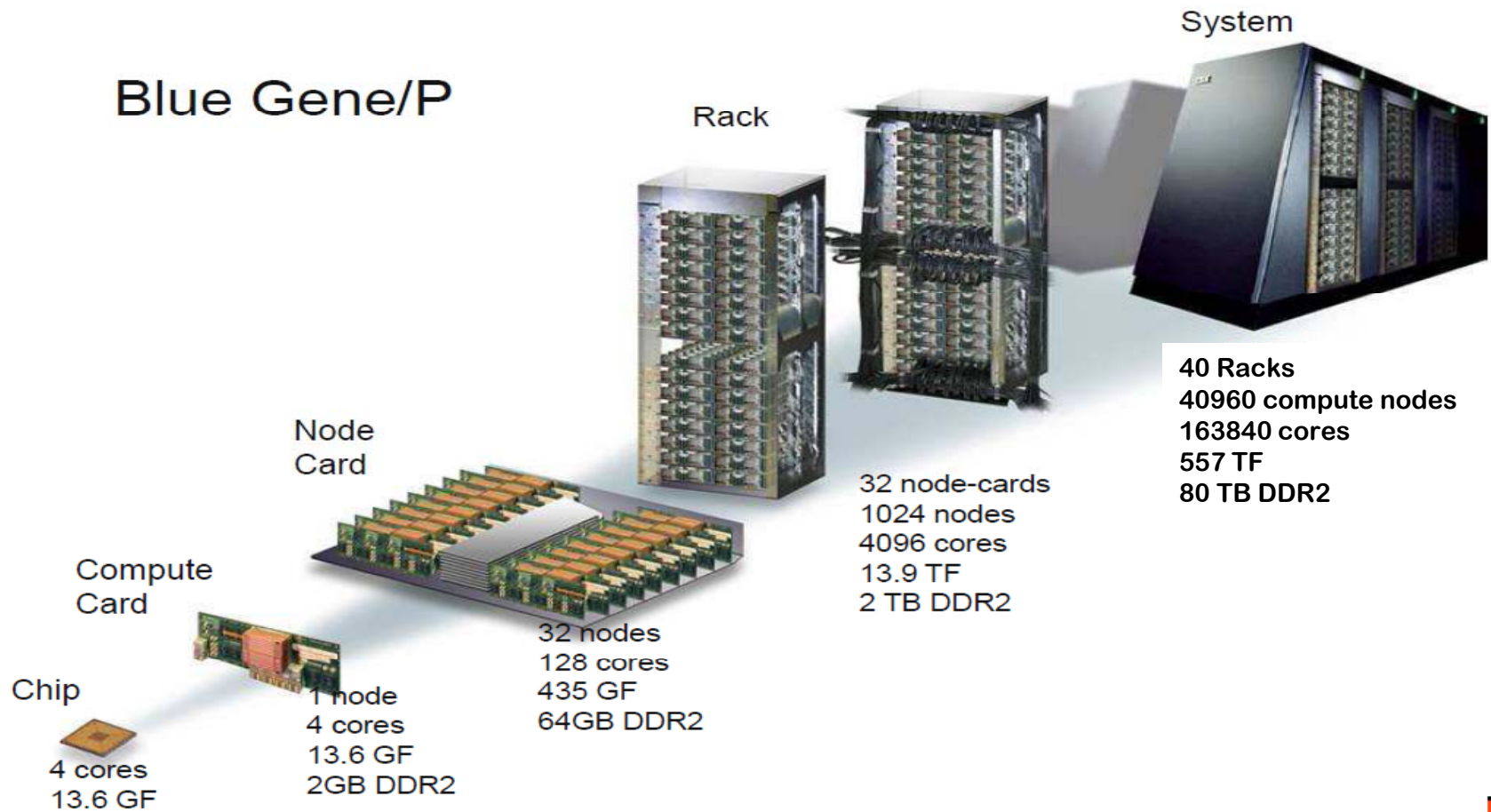
Evolution of the BG/P 'Intrepid' system

Argonne Leadership Computing Facilities

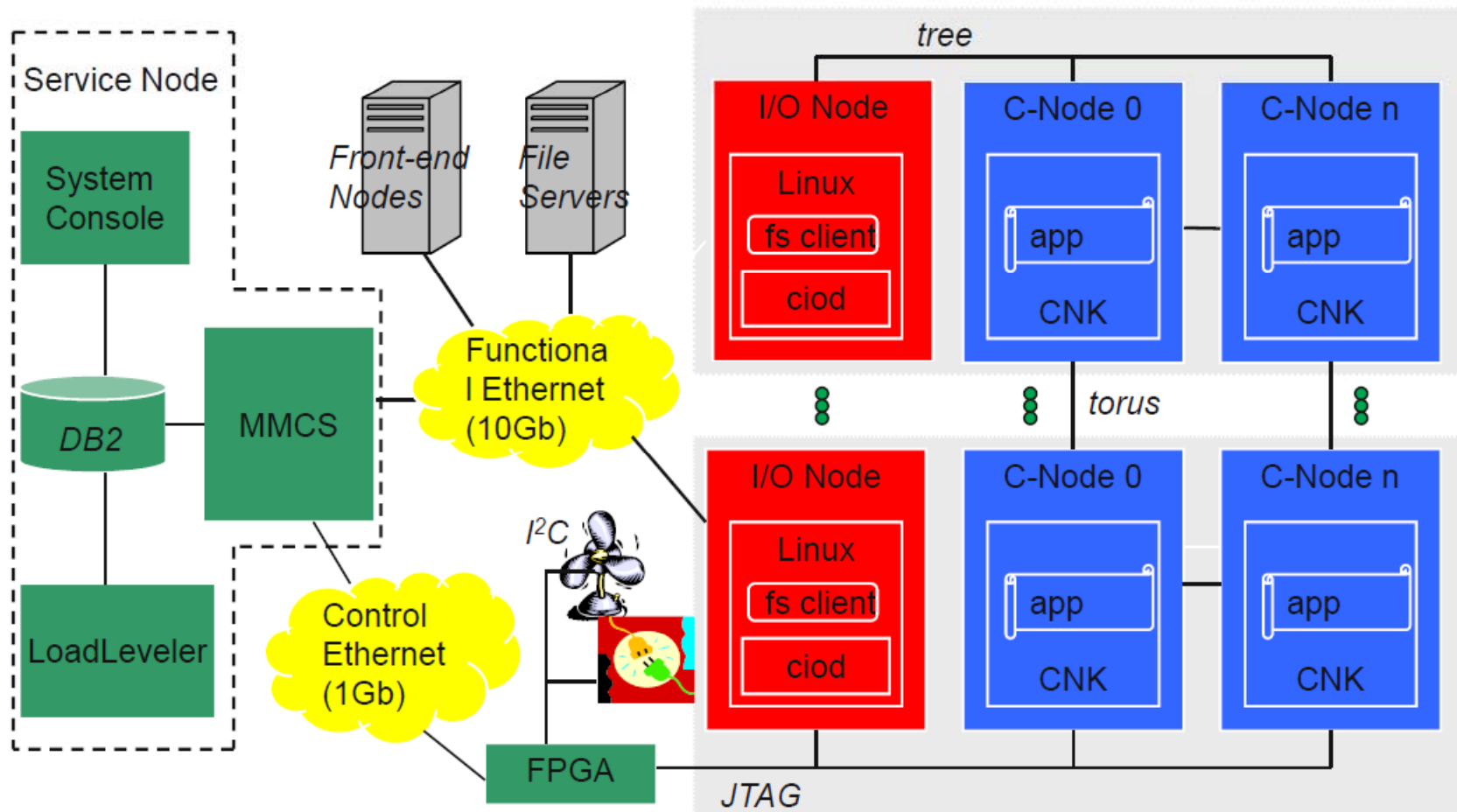
- Provides leadership-class computing resources to the scientific community
- The ALCF Blue Gene systems are open to DOE INCITE projects, and other select users
- **2004**
 - Formed of the Blue Gene Consortium with IBM
 - DOE-SC selected the ORNL, ANL and PNNL for Leadership Computing Facility award
- **2005**
 - Installed 5 teraflops, 1,024 dual-core Blue Gene/L for evaluation
- **2006**
 - Began production support .Continued code development and evaluation
- **2007**
 - Installed 100 teraflops Blue Gene/P *Intrepid* system (late 2007)
- **2008**
 - *Intrepid* system upgraded to a total 557 Teraflops
 - Currently, 40 racks with 40,960 quad-core compute node and 163,840 cores
 - Blue Gene/L retired in July 2008

Composition of BG/P machine

Blue Gene/P



BG/P System Architecture



Fault Tolerance in Intrepid

- Certain hardware on the Intrepid has redundancy incorporated
 - DDN backend storage has redundant controller, hot swapping, 8+2 RAID
 - Myricom 10-GbE network is redundant to a certain extent
 - If you lose spine cards, you still function but with loss of performance
 - If you lose leaf cards, attached nodes are lost
- The MMCS control system, which runs on the service node and manages booting, configuration and monitoring of all Compute and I/O nodes is very robust
 - Any error event detected in any part of the system hardware give rise to a *RAS* (Reliability, Availability and Serviceability), *event message*, which is sent to the MMCS server process and stored in a RAS database.
 - RAS database is a prime source of fault information on the system
 - The BG/P RAS architecture is more structured as compared to the BG/L; RAS messages handled more efficiently
- Additional logs generated by I/O nodes, Job scheduler, Myricom network and file system
- All components continuously monitored through custom scripts, IBM's BG navigation Navigator
- In addition, system is fully stressed during maintenance cycle every week

Fault Tolerance Challenges on Intrepid

- Certain hardware on Intrepid is not redundant
 - 10-GbE links from I/O node to Myricom network,
 - 10-GbE links from file servers to Myricom network; closely monitored
 - A single service node
 - Service node is a very robust enterprise class 32 CPU, 64GB RAM; closely monitored
 - All logs are stored on the Service node; backed on a tape drive
 - Hot spare is supported in new IBM drivers but not tested
- No RAS events are available for the Myricom network, file servers and DDN storage in the BG/P RAS infrastructure
 - Debugging issues can be tough; close monitoring needed for these components
- RAS information generated is enormous
 - Fatal events, warnings and information events
 - Not easy to keep track of all information; Can be difficult to perform root cause analysis
 - How to do co-relate job scheduler logs with I/O node logs to RAS logs

Most common errors seen on the Intrepid?

- Boot Failures
 - System partition is booted with a new copy of the operating system with every job
 - Involves mounting of a number of NFS, GPFS and PVFS file systems.
 - Boot failures are the result of network and GPFS issues that have not yet been 100% resolved
 - Latest fix from IBM has reduced the failure rate
 - Jobs used to fail in batches with long no-failure intervals
 - If your job starts (i.e.. the partition booted successfully), then the failure rate is very small (as mentioned in previous slide)

What are applications doing for FT?

- Applications use checkpoint mechanisms
 - Some more efficient than others
- Users tend to checkpoint based on the current perceived failure rate of the system and the length of the job
- **What else can applications do?**
 - Tricky question : Applications don't have enough information to do anything else
 - Perhaps can do validation, or run duplicate runs of a code with the same input files to compare results, check to see if the results look "reasonable"
 - Collaborate with folks doing FT research
- **What are application developer reluctant to do?**
 - Modify their algorithm or programming model, unless there is clear and very large benefit
 - Modify code if it will impact portability

How can the FT on such systems be improved?

- 2-way communication with vendor
 - Ex: Distributed manager (i.e. service node) to eliminate single point of failure
 - Deeper understanding of how the system is to be used; in order to understand the patterns seen in the field
 - What issues will be faced by next-generation systems
- Easier methods to predict faults, conduct root cause analysis, take automatic actions, reduce admin workload
 - Component count in high-end computing systems are increasing; thus failure counts are increasing
 - Maintaining a reliable system requires many many human hours
 - Future exa-scale system promise multiple ‘continuous failures’ each day
 - More research in the area of Fault Tolerance, focusing on these issues
 - FT in handled independently for each component; this ‘un-coordinated’ approach for FT is not going to work easily for big machines!

CIFTS Project Outline

- Motivation
 - Traditional Fault Tolerance is handled by individual components
 - Storage (RAID variations), File Systems (dCache, Tera Grid FS, Panasas, IBRIX, BulkFS)
 - Checkpointing software (application check-pointing ex: BLCR, Condor; operating system check-pointing ex: TICK)
 - Software built using hardware technologies like Imsensors, OpenIPMI, BMC
 - What's the Problem?
 - Virtually no coordination between different components
 - A component cannot inform other components about the faults it is seeing or benefit from the knowledge of faults that other components are seeing
 - Software don't know the reason for other system-wide faults
 - Did the application exit due to an inherent error?
 - OR
 - Did the application exit due to a hardware error?
- What does CIFTS provide?
 - An essential, much-required, **coordination framework** for different components to exchange hardware and software fault information

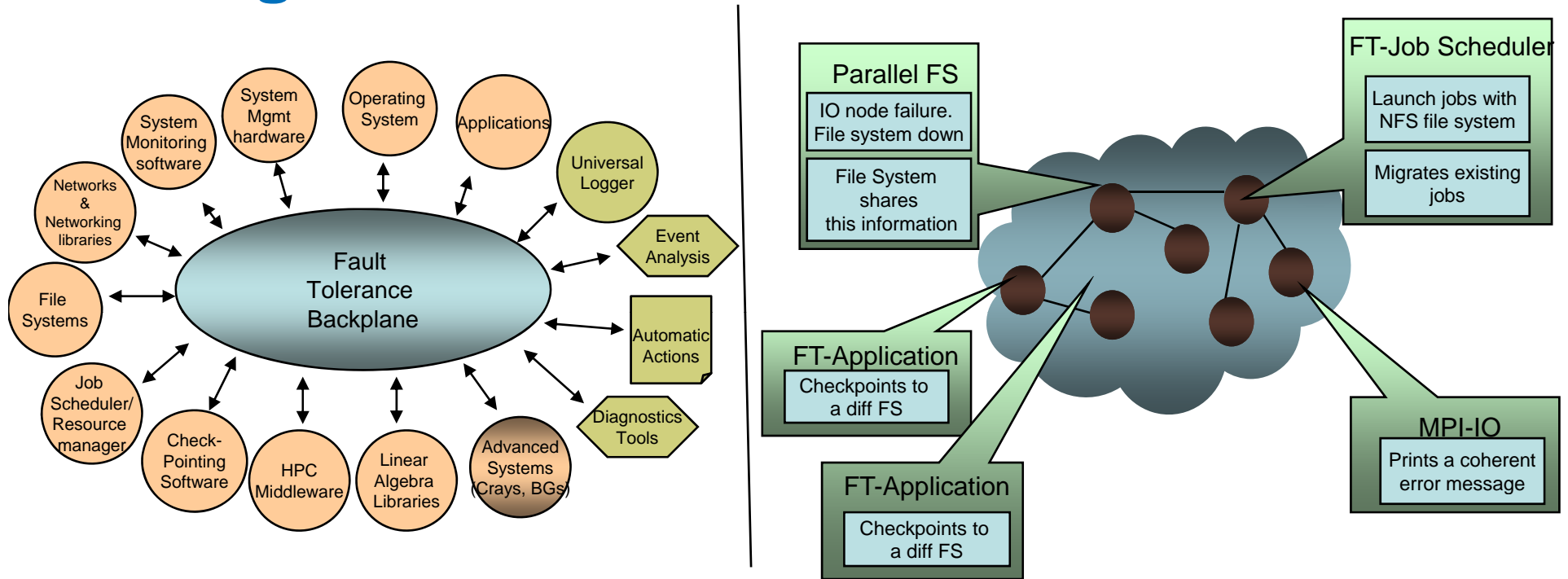
The Coordinated Infrastructure For Fault Tolerant Systems: Objectives

- CIFTS provides a **coordination framework** for different components to exchange hardware and software fault information
- Core of the CIFTS infrastructure is a **Fault Tolerance Backplane (FTB)**
 - FTB provides a scalable event handling and notification mechanism for all HPC software
- **Any software can utilize FTB to coordinate with other components**
- CIFTS provides software developers opportunities to improve their software's fault tolerance capabilities -- based on faults experienced by others on the system



The current CIFTS Team

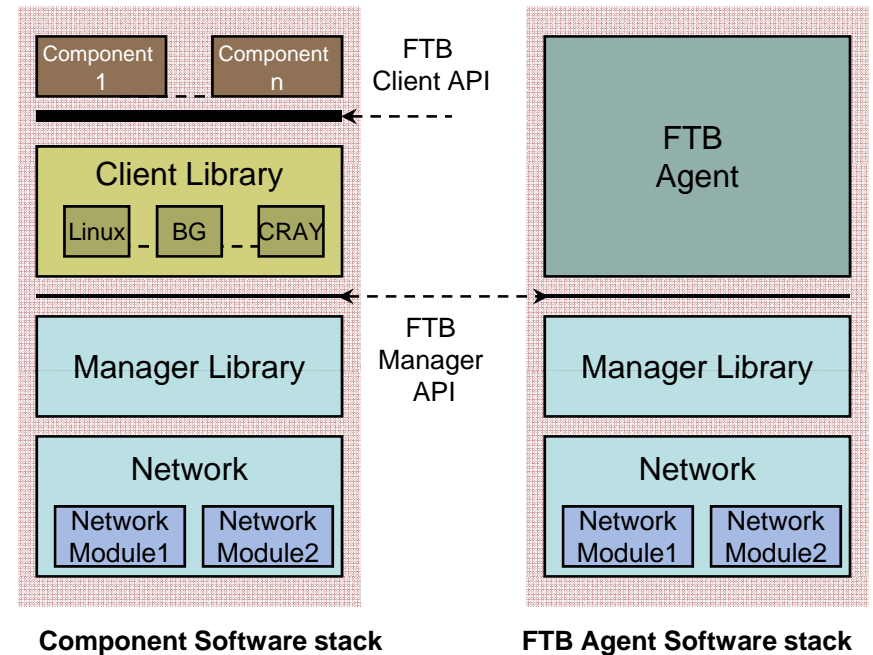
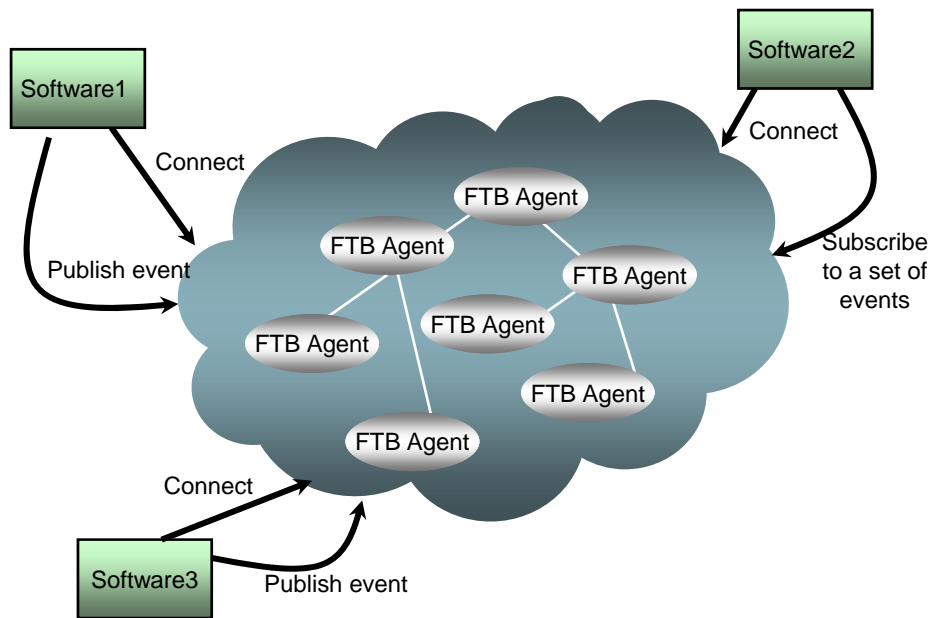
Fault Tolerance Framework & Scenario Using FTB-Enabled Software



The **Fault Tolerance Backplane** provides

- A scalable framework to exchange fault-related information
- Exposes a standard interface that can be used by any software to connect to the FTB
- Provides a common uniform event handling mechanism and event notification mechanism

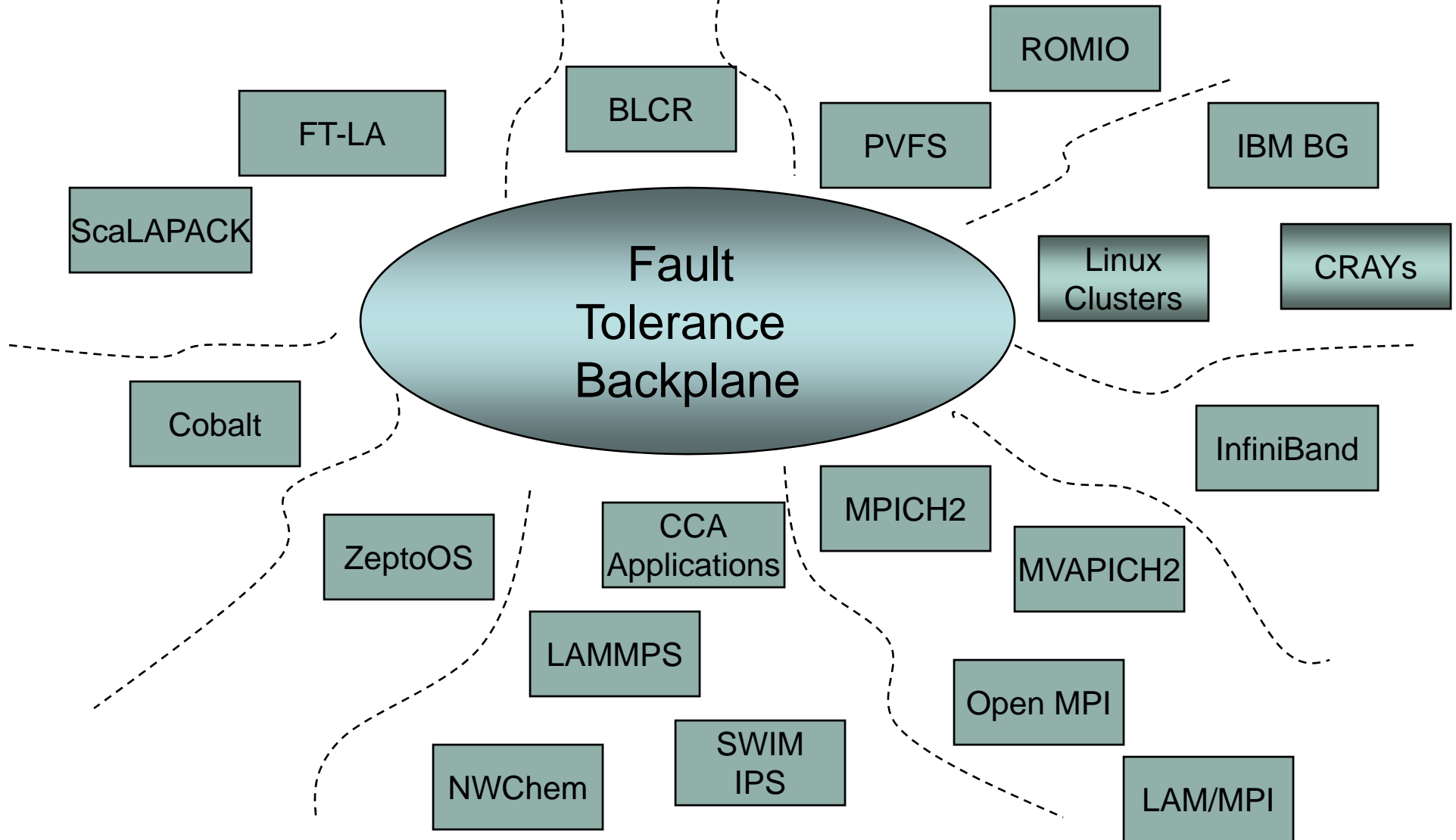
The FTB Architecture and API



FTB Client API -- Provides a set of simple FTB routines based on the publish-subscribe framework

- FTB_Connect()
- FTB_Publish_event()
- FTB_Subscribe()
- FTB_Poll_for_event()
- FTB_Unsubscribe()
- FTB_Disconnect()

FTB-enabled Software -- Planned



Accomplishments/Future Work with CIFTS

- **FTB 0.6 released in Sept 2008**
 - Platforms supported:
 - Cray XT
 - IBM BG/L and IBM BG/P running ZeptoOS
 - Linux clusters (tested with Ubuntu 8.04)
 - FTB API: Specified in the FTB developers guide
 - Download at <http://www.mcs.anl.gov/research/ciffts/software>
- **FTB-Enabled Software**
 - Ohio State University's Network-Based Computing Laboratory FTB-Enabled InfiniBand Network Monitoring Software (FTB-IB-1.0)
 - Download at <http://nowlab.cse.ohio-state.edu/projects/ftb-ib>
 - LBNL's Berkeley Checkpoint/Restart software (v0.8.0) for Linux provides technical preview
- **Vendor Engagements**
 - **IBM, Cray**
- **FTB 0.7 planned mid 2009**
 - Improvements to the inherent fault-tolerance of FTB
 - Distributed database with multiple distributed replicas
 - Topological improvements for fast restoration on faults (currently tree)
 - Event Coalescing (currently each event is handled separately)
 - Event Prioritization

Contacts

- Argonne Leadership Computing Facility
 - Mailing list: support@alcf.anl.gov
 - <http://www.alcf.anl.gov/>
 - Wiki: https://wiki.alcf.anl.gov/index.php/Main_Page
- CIFTS
 - Website : <http://www.mcs.anl.gov/research/cifts/>
 - Mailing list: cifts_discuss@googlegroups.com
 - Contacts
 - Rinku Gupta (rgupta@mcs.anl.gov), Technical Lead
 - Pete Beckman (beckman@mcs.anl.gov), PI