Satisfying Operational Requirements for Fault-Tolerance of NWP Ensembles

George R Carr Jr
Thomas B Henderson

General Motivation

- Operational goal to reduce the probability of a failure of the forecast cycle (production, time critical processing)
- Recent and planned changes increase the probability of failures
  - Desire to increase in resolution
  - Desire to increase in number of ensemble members
  - Additional science and changes in the implementation of the science
- Provide a method to reduce the probability of a complete forecast failure while facilitating "new science"

Goal of this activity

- Define a use case target
  - Which issues can you afford to fix (time, effort, complexity, etc.)?
  - Not addressing "Exascale" fault issues. Application/model level target.
- Provide a proof of concept demonstration of a mechanism capable of satisfying the use case
- The solution for the general ensemble forecast use case should be
  - Flexible enough for additional use cases
  - Simple, well understood, portable, maintainable mechanism
  - Easy to incorporate into ESMF (Earth System Modeling Framework)
- Provide a new level of Fault Tolerance: detect, prune, continue

Limited scope of this work

- Does not explicitly address
  - "Exascale" although this will be one aspect that will continue into the Exascale era
  - Hardware
    - Power
    - Storage
    - Network
  - Software
    - O/S
    - Job scheduling
- Does address
  - Application level fault tolerance due to hardware or software issue at the node level

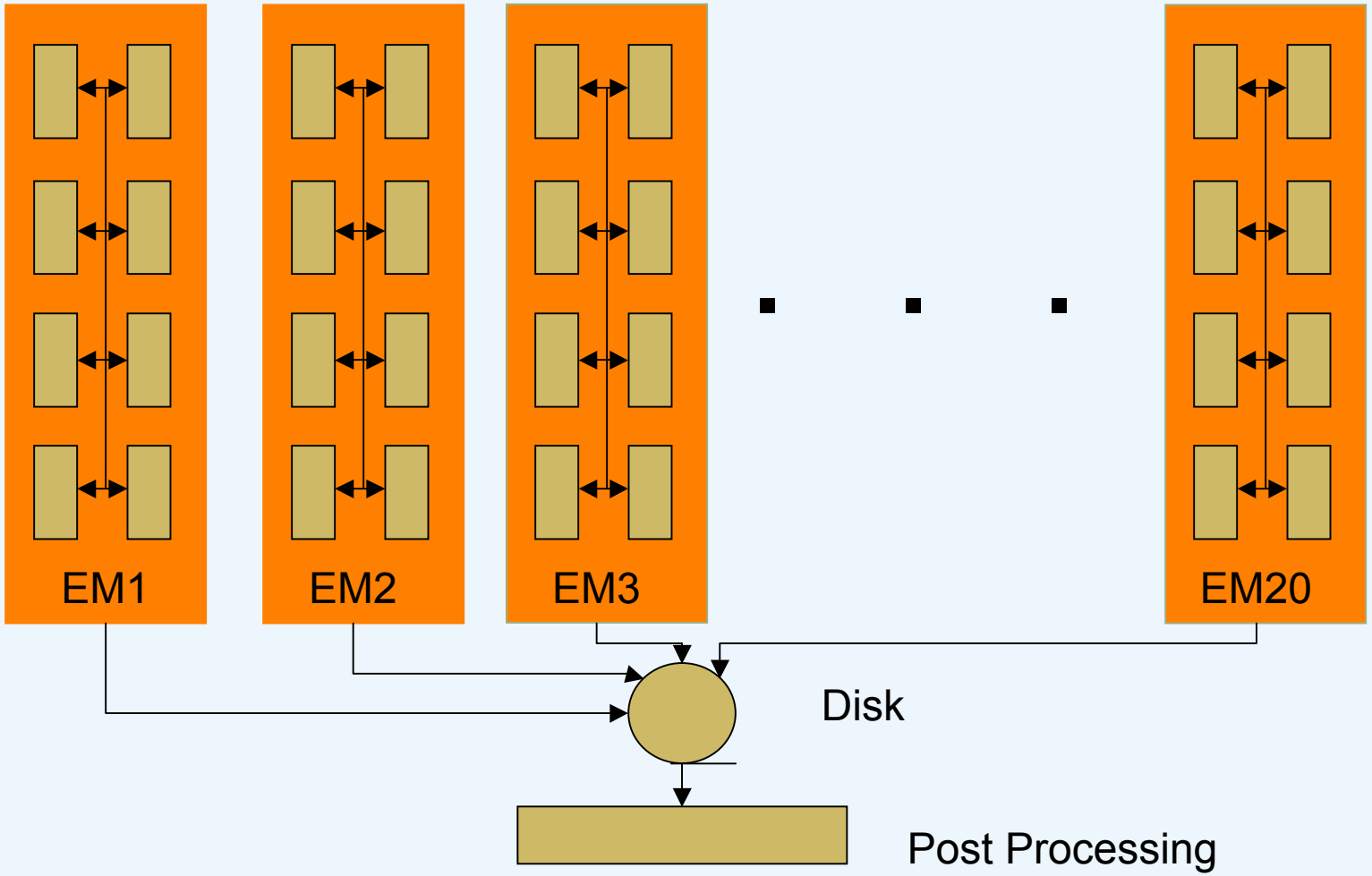
Near term science motivation

- New science
  - Every N time steps all ensemble members need to communicate current processing status for evaluation.
  - Processing parameters may be adjusted on the fly, mid-stream during the forecast job.

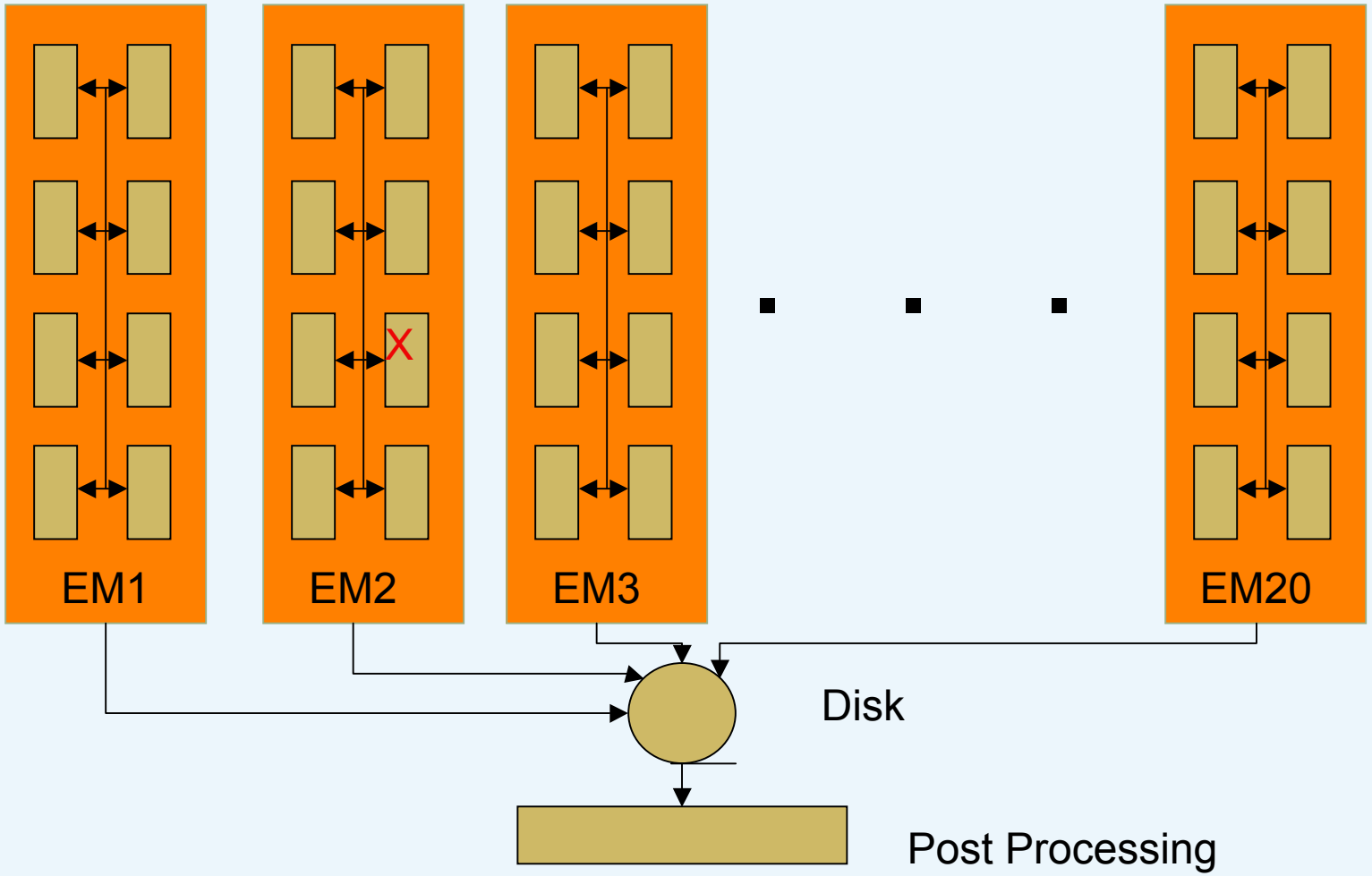
Operational Use Case

- Fault-tolerance, not fault-recovery per se - upon a failure, permit surviving ensemble members to continue and complete
  - Addressing deadline processing concerns (not real real-time processing)
  - Must be able to set max timeout for ensemble member failure declaration
  - Prune and continue

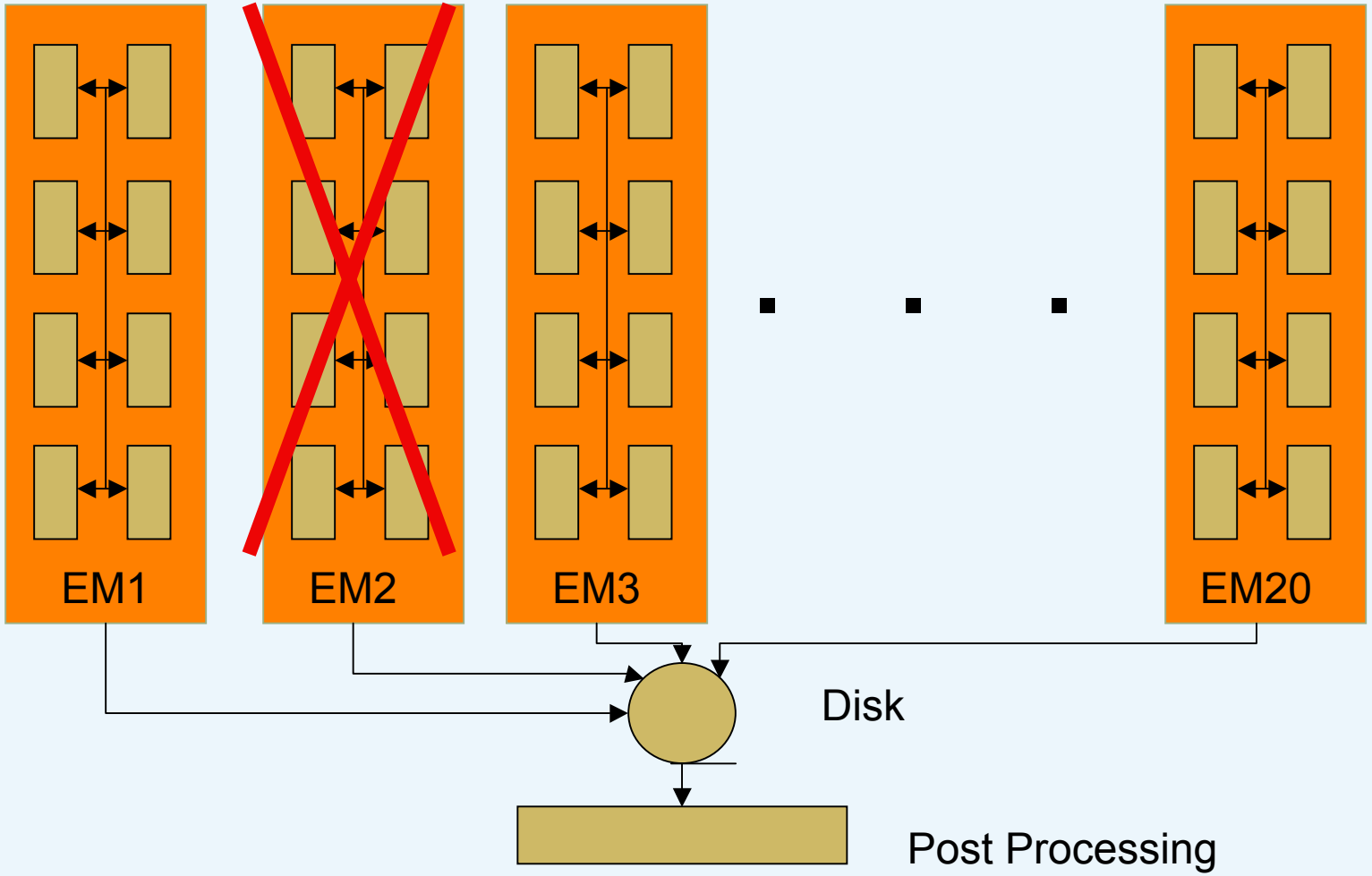
Traditional Ensemble Forecast



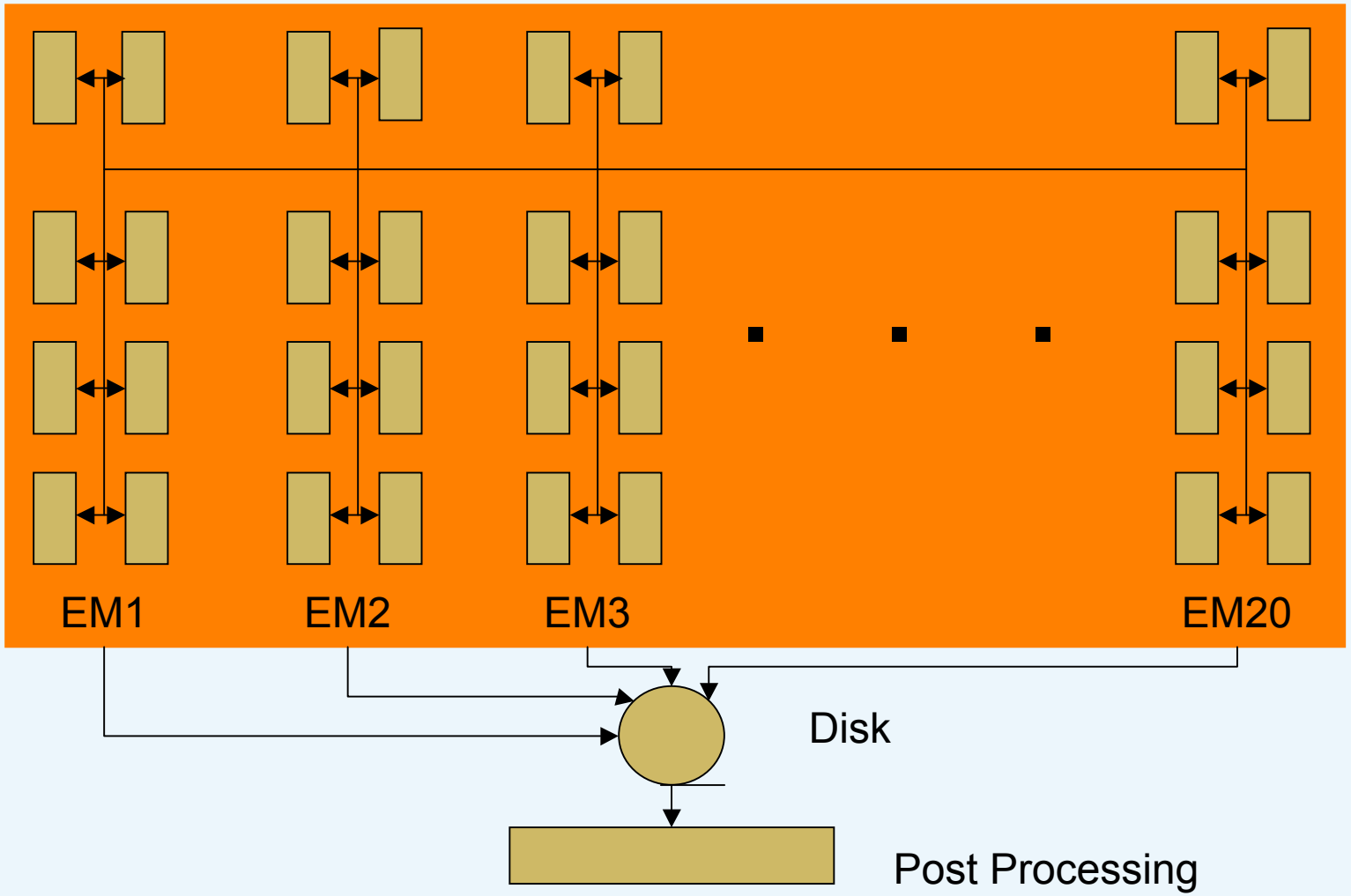
Traditional - A Single Failure



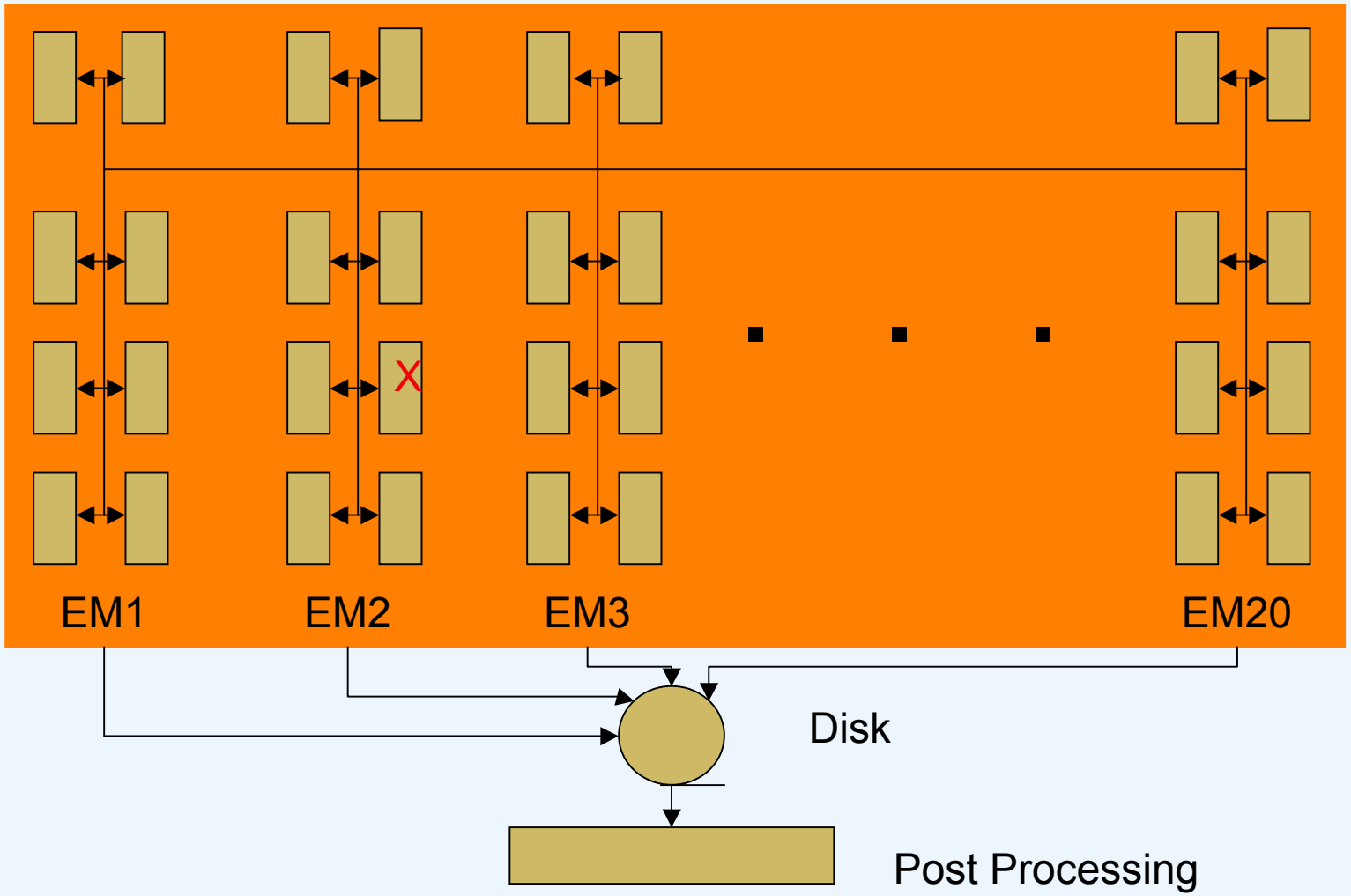
Traditional - Failure Result



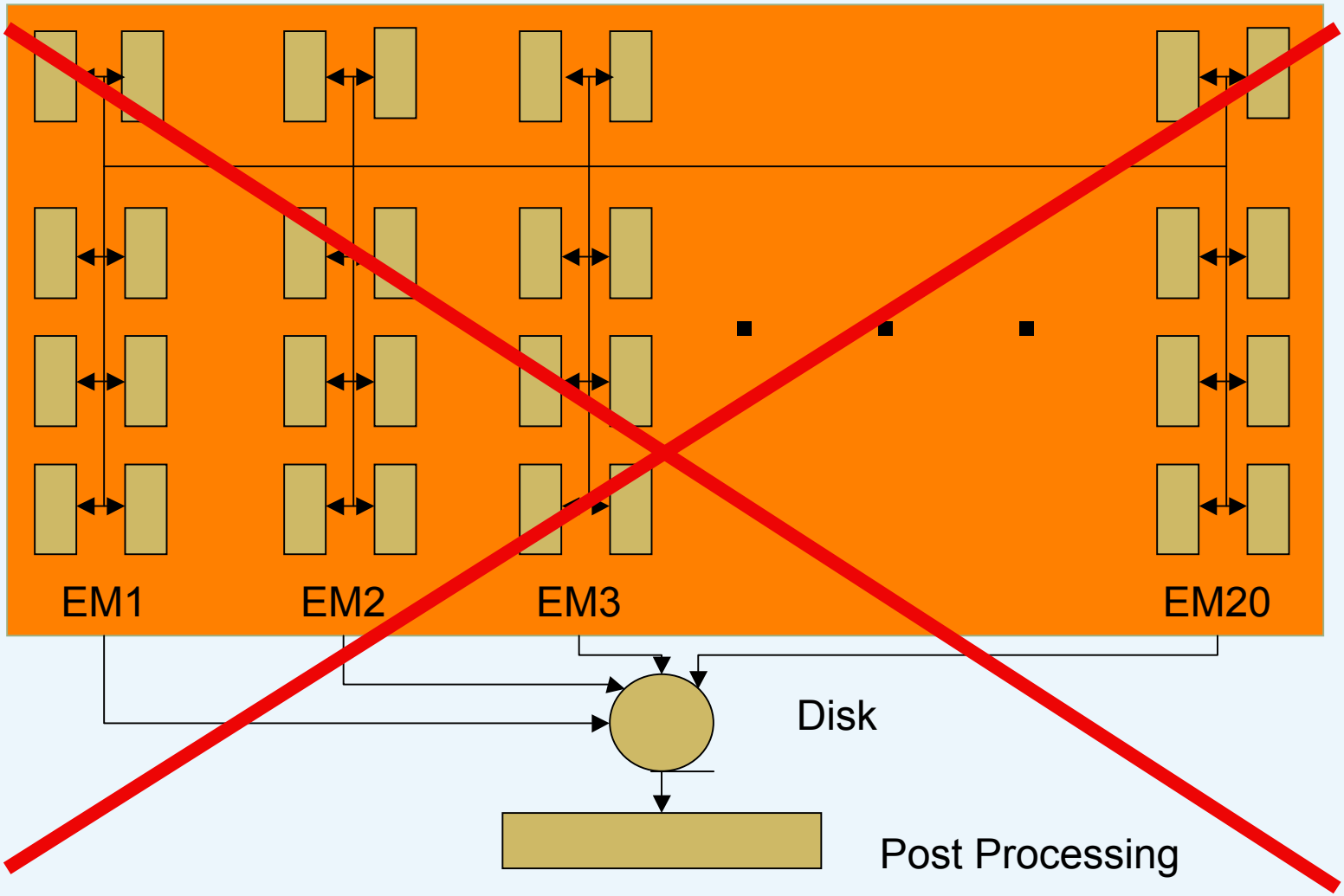
Current Ensemble Forecast



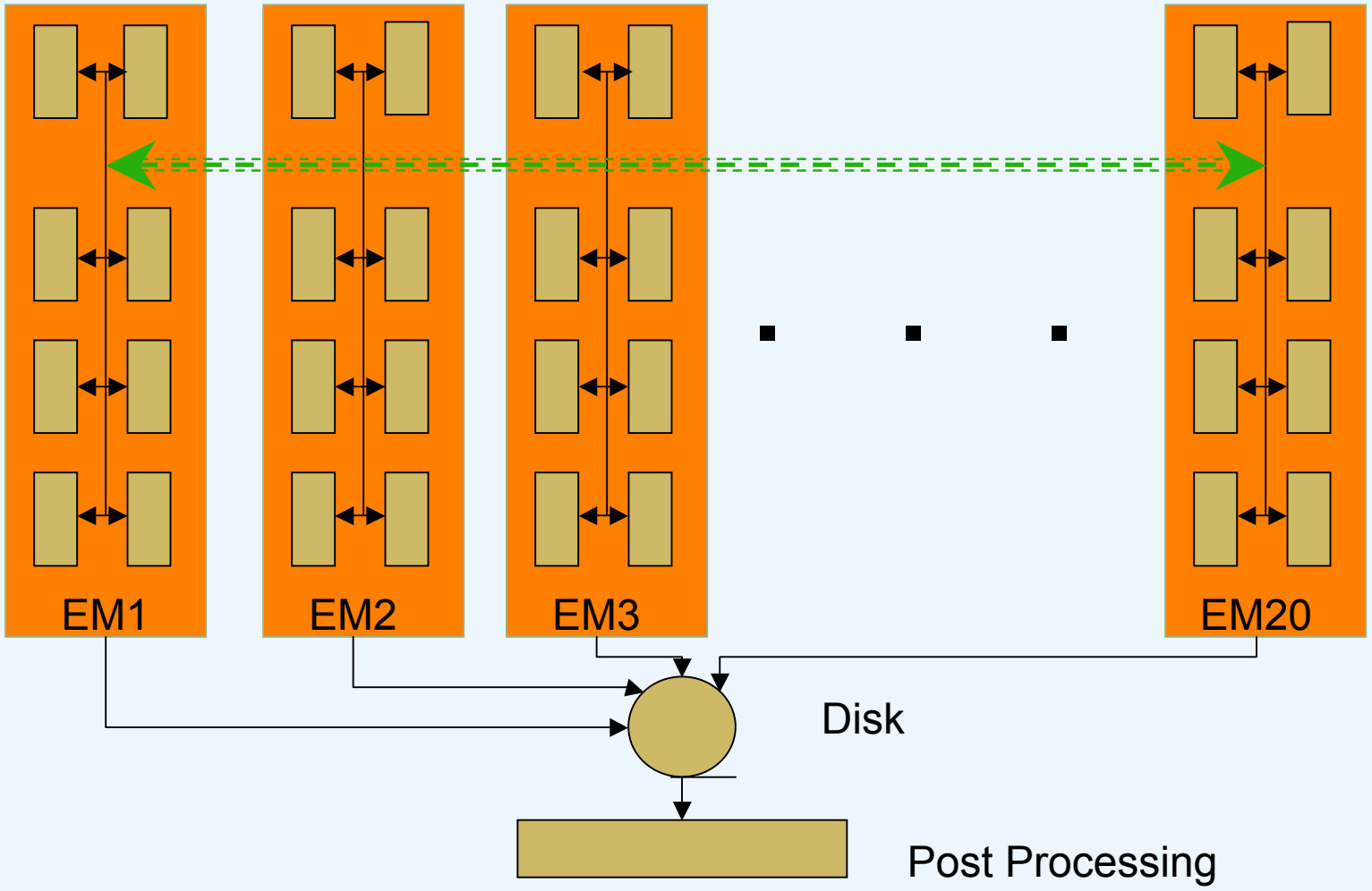
Current - A Single Failure



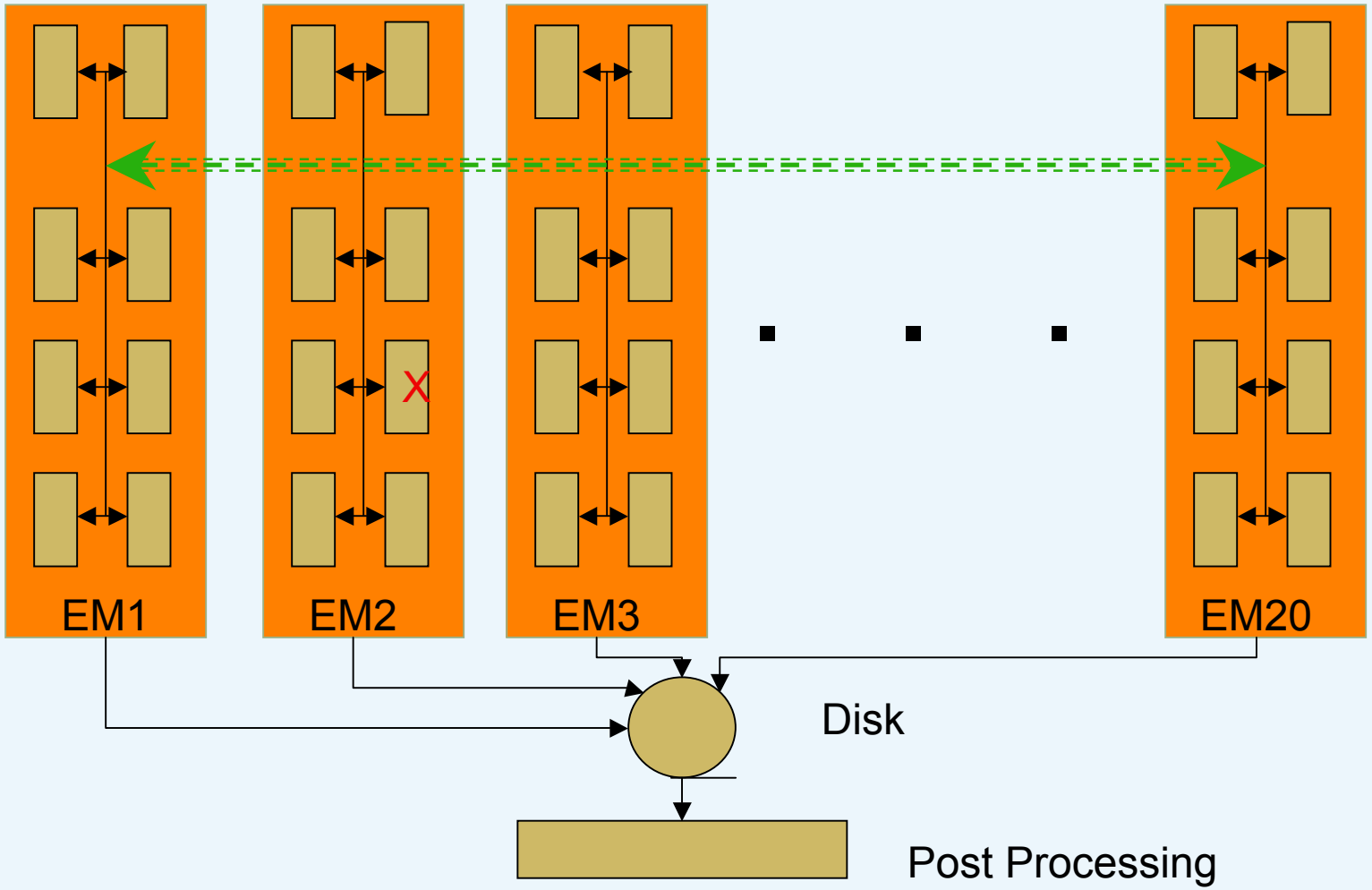
Current - Failure Result



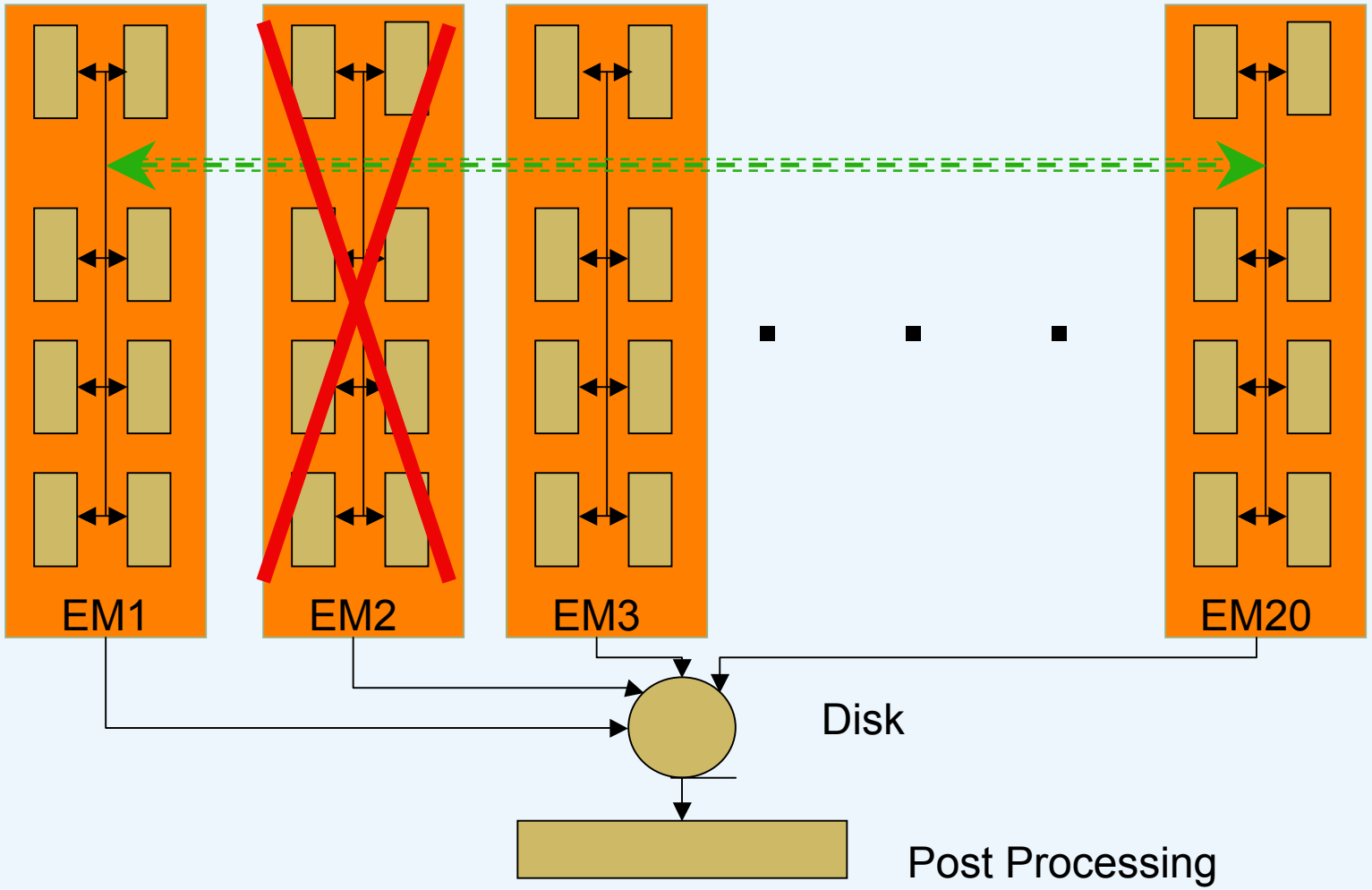
What is needed?



Future - A Single Failure



Future - Failure Result



Consider A Bicycle Team Time Trial

(a la Le Tour de France)

- 9 riders start the race as a team
- All riders (still in the race) work together to complete the race
- A problem with A bike or A rider (hopefully) only results in the loss of that rider
- If a team leader fails, another takes over (all somewhat equivalent)
- Time is still a limiting factor (must finish within a max time window)
- A minimum number of riders must finish for the result to matter

- Consider the team to be an ensemble forecast.
- Consider each rider to be one of the forecast ensemble members.

Possible Approaches

Mechanism	Comments	Portability	Additional Maintenance / Support	Ease of Use	Fault capable
Sockets	Built into Unix / Linux systems. Many others built on top of Sockets	Yes	None needed	OK. Some start up for most.	Yes
CORBA	Built on Sockets	Some	Some required	Object oriented targeting C++ and Java	Yes
DDS	Built on CORBA. More complexity than needed.	Limited	More required	Object oriented targeting C++ and Java	Yes
Disk based	Coordination via shared files requiring locks and synchronization	Yes	Some required	Configuration scriptable. Code required.	Somewhat. Some failure modes would prevent continuation.
PVM	Older not widely used anymore	Mostly. Some updating might be needed	Last release in 1993	Was a niche player, replaced by MPI-1	Yes
MPI2	Well known and used standard	Yes	None needed	OK. Well understood.	No

Event Sequence

- Each ensemble member would:
  - Initial rendezvous: initialize fault-tolerant communication. Members M[1..n-1] register with M[0]
  - Initialize model
  - Primary Work Loop
    - Perform model computation
      - For failure testing, caused numeric exception on specific iteration
    - Every specified number of model time steps, perform a rendezvous
      - Current “master” waits for N-1 others to connect. Timeout those that do not within time window. Trigger new science processing. Ack to “workers” with current member list and results of new science.
      - “Workers” send new science data to current “master”, wait for Ack. If timeout then failover and rendezvous with new “master”
  - Final model work
  - Model termination
  - Terminate fault-tolerant communication

How It Works

- Each ensemble member is a separate but orchestrated MPI launch.
- Ensemble member zero starts out as “current master”
- Any ensemble member task failure takes out only that ensemble member.
- Only one ensemble member at a time is the “current master”. At each rendezvous, the “current master” provides all “remaining” members a list of the “remaining” members which is used for succession.
- If a member fails to rendezvous, the “current master” will declare it gone and proceed with the remaining members.
- If the “current master” fails to respond, the other members will declare the next member on the list to be the new “current master” and proceed.
- A parameter value sets the maximum time to wait for an ensemble member to rendezvous.
- “Master” and “Worker” might be identical binaries, just execution flow differences. All ensemble members capable of becoming “Master”.

Consider the following

- A memory or other hardware error results in failure of a node (must allocate whole nodes to one ensemble member)
  - Result: Prune the ensemble member using that node and allow the others to continue
- A solver has an issue with the specific data resulting in a software exception
  - Result: Prune the ensemble member using that node and allow the others to continue
- Suppose a network switch fails breaking the ensemble in half
  - Result: it might be possible that each half continues without the other and completes (possible issue I/O)

Solution Advantages

- Simple yet sufficient architecture
- Portable, maintainable underlying transport mechanism
- Extensible to possible future constructs
  - Multiple models in a single forecast ensemble
  - Multiple machines in a single forecast ensemble
- Would support hybrid cpu/gpu computational platforms
- Socket is a common mechanism used by MPI and the one used by ESMF

An Alternative Future: MPI-3

- MPI Forum
  - Bi-weekly conference calls
  - Bi-monthly meetings
  - NWP use case was presented
  - Formal first draft presented at July meeting, passed just a couple weeks ago [FT WAS DROPPED]
  - Standard addresses additional items, some of which NWP might want/need in the future
  - Early beta implementation supports early testing
  - Vendor participation will help guarantee widespread adoption

MPI Forum Fault Tolerance Working Group: User-Level Failure Mitigation (ULFM) Run-Through Stabilization (RTS) Proposal

– or –

Keeping MPI applications running even when one or more processes fail during execution

Joshua Hursey
Assistant Professor
Computer Science Department
University of Wisconsin-La Crosse
jjhursey@cs.uwlax.edu
http://cs.uwlax.edu/~jjhursey/

Message Passing Interface (MPI) & Fault Tolerance

- **High Performance Computing's need for Fault Tolerance**
  - Large scale and long runtimes lead to increased opportunity for failure to disrupt the application run.
  - Process failure will become a normal event for exascale apps.
  - Checkpoint/restart techniques alone will not be enough
    - Need to look at Natural & Algorithm Based Fault Tolerance (ABFT) e.g., volatile groups of processes (ensembles), checksums stored in peers, rewinding computation, redundant computation, …
- **Entire HPC software stack lacks enough support for portable, fault tolerant applications.**
  - Projected that we need a fault tolerant MPI by 2012 in order to support the development of fault tolerant applications by 2016.
  - MPI is a critical piece of software for most HPC applications
    - However it cannot withstand process failure in a portable manner

Dongarra, J., Beckman, P., et al., "*The International Exascale Software Roadmap*," International Journal of High Performance Computer Applications, 2011.

Fault Tolerant MPI Proposal: User-Level Failure Mitigation (ULFM RTS)

Define a set of semantics and interfaces to enable fault tolerant applications and libraries to be portably constructed on top of MPI

- **Application involved fault tolerance (not transparent FT)**
  - Application has to opt-in to use this feature (backwards compatibility)
- **Fail-stop process failure** (process crash)
  - A process failure in which the MPI process permanently stops communicating with other MPI processes, and its internal state is lost.
- **ULFM RTS Proposal allows an application to**
  - Receive notification of process failure,
  - Continue execution with the remaining processes,
  - Rebuild or discard communicators as needed,
  - Reach agreement among all alive processes,
  - Replace failed processes (currently via MPI_Comm_spawn)

Fault Tolerant MPI Proposal: User-Level Failure Mitigation (ULFM RTS)

- **Availability:**
  - MPI Standard:
    - The proposal missed the deadline for MPI 3.0 (due out Fall 2012).
    - It is under consideration for the next release of the MPI standard.
  - Prototypes:
    - Open MPI Public Beta: Full implementation available at http://www.open-mpi.org/~jjhursey/projects/ft-open-mpi/
    - MPICH: Partial support in MPICH2 trunk
    - Other (vendor) implementations: Some are in progress at the moment
- **What we need:**
  - Before this proposal can be standardized the MPI Forum needs to see demonstrated practice.
    - This means that we need to show real applications using this interface
    - If you willing to experiment with the interface please get in touch with the Fault Tolerance Working Group.

MPI-3 Observations

- Original FT proposals covered many of the things we need
- "FT lite"
  - There were concerns within the Forum regarding what could reasonably be adopted, implemented and supported.
  - Less deterministic behavior
    - How/where will faults be reported
    - Missing even a loose timeout (we don't need hard realtime-ish but …)
    - Eventually an error will be posted but each MPI member may receive a different error code.
- FT deleted from final MPI-3 draft standard (September, 2012). The FT working group will meet at SC and in December to consider options.
- MPICH (ANL) and OpenMPI (ORNL) alpha/beta implementations in development
- Goal: Multi-vendor, stable, productive interface for users including Exascale class problems

- My guess: min 2 years before we have first stable product

Project Status: Complete

- Requirements understood for first operational use case
- Initial design completed
- Initial proof of concept code written and tested
- NWP use case presented to MPI Forum
- Preliminary capability encapsulated and added to ESMF
  - See ESMF Fortran User's guide, Chapter 17. R5.3.
  - http://www.earthsystemmodeling.org/

Future Activities

- May include:
  - Building a demonstration using one of the NWP ensembles
  - Working with and/or monitoring MPI Forum to keep NWP needs in consideration for future MPI Forum specifications
  - Examination of and/or early testing of MPI-3 beta releases

Special Thanks

- Funding:
  - NOAA/GSD
  - NCEP
- Support
  - Joshua Hursey, U of Wisconsin (formerly ORNL) - an active participate of the MPI Forum
  - Gerhard Theurich, NOAA - ESMF
  - Tom Henderson, NOAA/GSD
  - Leslie Hart, NOAA/CIO
- ECMWF for hosting this workshop

Thanks
Any questions?

George.Carr@noaa.gov