Three Crazy Ways to Cope with Failure that Will Change Your Apps Forever

Salishan Conference
April 30, 2015

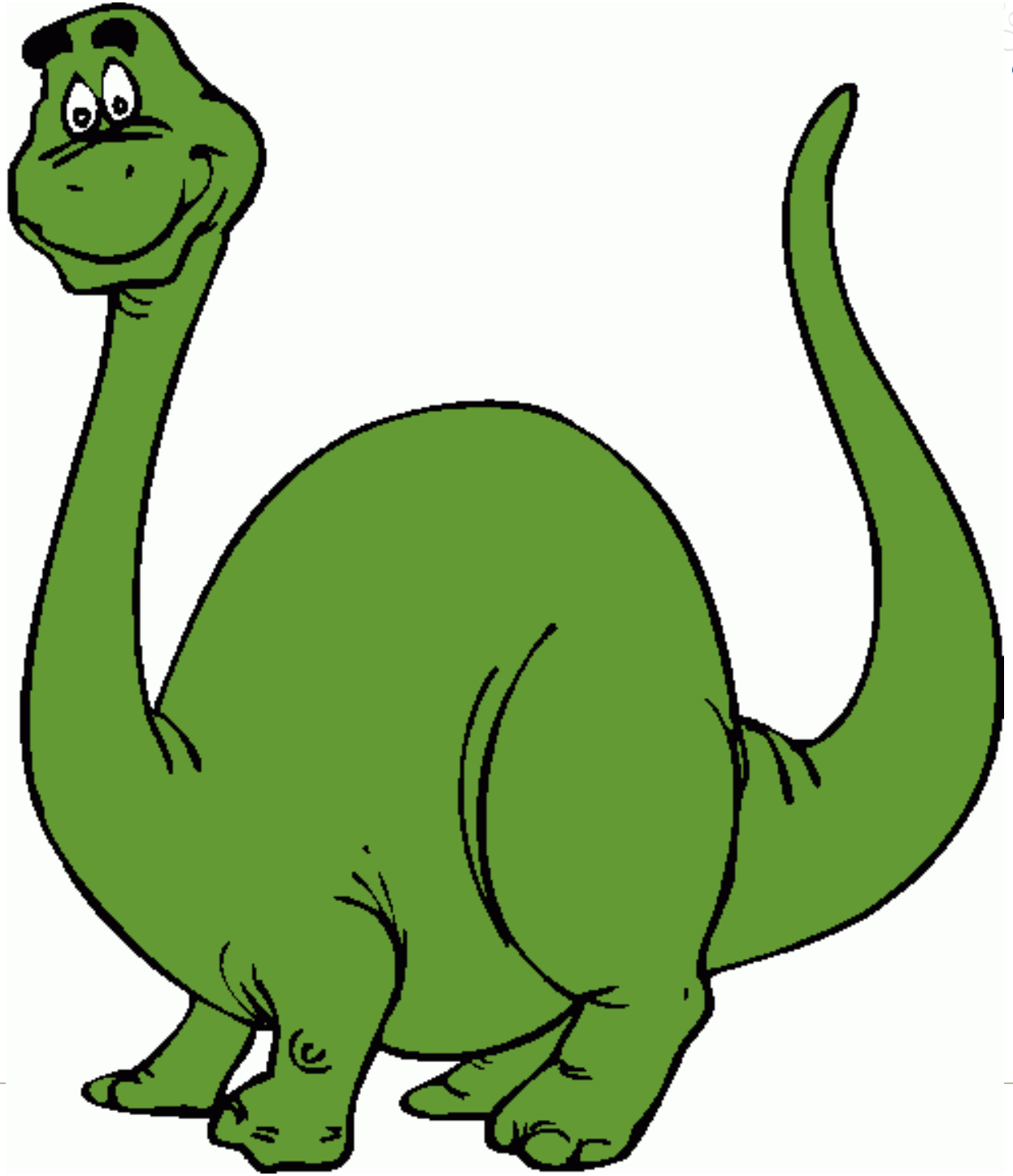
Sung-Eun Choi, Principal Engineer
Cray Inc.

Not So Three Crazy Ways to Cope with Failure that Will Change Your Apps Forever

Salishan Conference
April 30, 2015

Sung-Eun Choi, Principal Engineer
Cray Inc.

RAWR!



Never say die

- **Certain parts of your system do everything they can to never do the wrong thing**
- **Hardware is full of error handling capabilities**
  - See Session 1 talks
- **So is the OS**
  - e.g., Linux vfs_read() ~60% source code is for handling errors
- **Mature middleware is less so, but still pretty good**
  - e.g., Most MPICH calls have an error exit path


BRACE YOURSELVES
EXASCALE IS COMING

Crazy Idea #0: Admit you have a problem

- **Five randomly chosen DOE mini apps**
  - SLOC devoted to error handling can be approximated by zero
- **But you can't blame apps**
  - MPI spec says nothing is guaranteed after return with error
  - OpenSHMEM doesn't have return codes
    - Maybe they're just being more realistic
- **But you can't blame communication libraries either**
  - Any error they can't handle themselves, no one really could… until recently

Outline

- Not So Crazy Idea #1
- Not So Crazy Idea #2
- Not So Crazy Idea #3

NSCI #1: Write a portable program

Brief interruption

I’m not here to tell you what programming model to use

This is a very personal decision

NSCI #1: Write a portable program

- **Fault tolerance solutions are going to be tied to the architecture (duh) and system software**
- **Implementing, testing and/or debugging is not *to scale***
  - e.g., NERSC Cori phase 1 comprised of Haswell processors, not KNL

Portability is more important than ever

NSCI #1: Resource management

- **Cray Environment:**
  - aprun launcher will restart the (shrunken) job after node failure
- **Slurm:**
  - New options for fault tolerance
  - e.g., add a node OR give me more time

Workload managers are part of the fault tolerance solution

But they're currently not part of the portability solution

NSCI #1: Challenges

- **There are a few different workload managers**
  - And they keep changing the options


I have altered qsub.
Pray I do not alter it again.

NSCI #1: Challenges

- **There are a few different workload managers**
  - And they keep changing the options
- **Workload managers are often customized for a site**
  - Policy implementations, preferred terminology, etc.
- **Can we find common idioms?**
- **Can we standardize?**
- **???**

NSCI #1: Recipe for success

Application programmers:

Write a portable program

Software providers:

Make resource management part of your portability story


PLEASE
tell me about your lock-free algorithm

NSCI #2: Understand your memory model

- A *Memory Consistency Model* (MCM) defines the order in which memory operations appear to execute

- Dekker's algorithm (1965-ish)

thread 1 … thread 2

sequential consistency

```
flag1 = 0xE;
if
}
```

…the result of any execution is the same as if the operations of all the processors was executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.

- This stopped working on most CPUs ~30 years ago
- Today there are half a dozen ways to make this work
  - On x86 most are equivalent from a performance standpoint

Brief interruption #2

If you have a couple hours to kill…

Watch Herb Sutter's talk

atomic<> Weapons:
The C++11 Memory Model and Modern Hardware

NSCI #2: Relaxed consistency

- **Compilers and hardware are conspiring run a completely different program than the one you wrote**
- **Relaxed consistency is about exposing yourself their trickeries, in exchange for performance (maybe)**
- **Herb Sutter's advice (my words):**

  Don't use relaxed consistency unless you're special

- **This community is special**
  - If there's something that can improve performance, they'll try it

NSCI #2: Should you care?

- **MPI rank on every core: No**
  - There are people working on reducing the overhead of doing this
- **MPI+X: Probably**
  - Depends on X and how you use it
- **PGAS/APGAS/SHMEM: Yes**
  - In fact, you've been caring for a while
- **Dynamic task-based programming models: Probably**

The runtimes of all these care

NSCI #2: Do I need to care for fault recovery?

- **If *yes* on previous slide, then yes**

- **SDC recovery techniques can access shared data**
  - Depending on your programming model, you may have to do additional synchronization
  - Or use techniques that can happen at existing synchronization points

NSCI #2: Example

- One-sided programming models decouple synchronization from communication:

```
shmem_put64(dest0, src0, len, pe);


shmem_put64(dest1, src1, len, pe);
```

NSCI #2: Example

- **One-sided programming models decouple synchronization from communication:**

```
// When is this put complete at pe?
shmem_put64(dest0, src0, len, pe);

// What about this put?
shmem_put64(dest1, src1, len, pe);
```

NSCI #2: Example

- **One-sided programming models decouple synchronization from communication:**

```
// When is this put complete at pe?
shmem_put64(dest0, src0, len, pe); // I have no idea!

// What about this put?
shmem_put64(dest1, src1, len, pe); // Me neither!
```

- **If you care, say for SDC, synchronize manually:**

```
shmem_put64(dest0, src0, len, pe);
shmem_put64(dest1, src1, len, pe);
shmem_quiet(); // Both puts guaranteed to be visible
```

NSCI #2: Challenges

- **MCMs are hard**
  - If anyone in the audience can explain `memory_order_consume`, please see me afterwards
- **It's not always clear you get benefit from using a more relaxed model**
  - e.g., change communication characteristics

NSCI #2: Recipe for success

Application programmers:

Understand your MCM (if you need to)

Software providers:

Specify a MCM that can be implemented efficiently

NSCI #3: Exceptions

NSCI #3: Exceptions

- **Don't (at least for a language like C++)**
  - Difficult to write good exception handling code (across threads?)
  - Not practical
- **See Google's C++ Style guide:**
  - https://google-styleguide.googlecode.com/svn/trunk/cppguide.html#Exceptions

Summary

NSCI #1: Write a portable program

- Portability continues to be important
- Resource management needs to be part of the portability story

NSCI #2: Understand your MCM

- An MCM is your friend, use it judiciously

NSCI #3: Exceptions are impractical

- Don’t use them (or rewrite all your software in Java)

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.