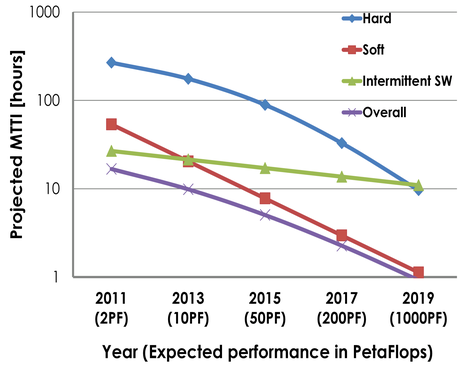
Containment Domains for Scalable and Efficient System Resiliency

Michael Sullivan, Jinsuk Chung, Dongwan Kim, Ikhwan Lee, Jee Ho Ryoo, Doe Hyun Yoon, Mattan Erez
The University of Texas at Austin

Introduction

Errors at all levels will become more prevalent in future systems.



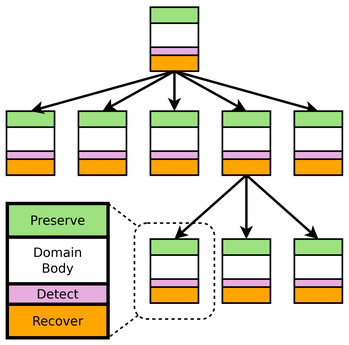
Anticipated error trends.

Current reliability techniques are typically have a fixed, high overhead. Issues that **must be addressed**:

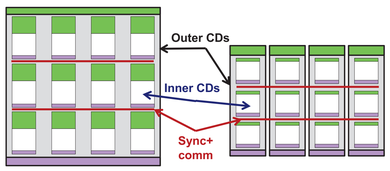
1. The requirements of an error detection and correction system are **application specific**
2. The storage hierarchy exhibits a large amount of **natural redundancy**; this redundancy can be utilized to avoid much of the cost of state preservation

Containment Domains

Containment domains (**CDs**) are a programming construct with weak **transactional** semantics that can be nested to take advantage of the machine hierarchy and to enable **distributed** and **hierarchical** state preservation and restoration. In addition, CDs allow the programmer to **tune and specialize error detection and recovery** to suit application needs.



(a) Hierarchy and Components



(b) Relaxed CDs

Scalable System Resiliency

Memory and disk bandwidth are scarce in current machines, and will become increasingly limited in the future. CDs can **scale to larger system sizes and higher error rates** by:

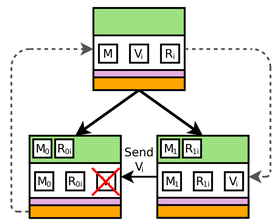
1. Providing resiliency at granularities which suit error rates
2. Exploiting natural redundancy within the machine
3. Effectively mapping to the storage hierarchy
4. Recovering locally in an uncoordinated manner when possible

The **flexibility** and **transparency** of CDs allows for an exploration of a rich set of tradeoffs involving **when**, **where**, and **what** data to preserve.

Example: Sparse Matrix-Vector Multiplication

```
void task<inner> SpMV( in matrix, in vec_i, out res_i) {
 cd = create_CD(parentCD);
 add_to_CD_via_copy(cd, matrix, …);
 forall(…) reduce(…)
   SpMV(matrix[…],vec_i[…],res_i[…]);
 commit_CD(cd);
}

void task<leaf> SpMV(…) {
 cd = create_CD(parentCD);
 add_to_CD_via_copy(cd, matrix, …);
 add_to_CD_via_parent(cd, vec_i, …);
 for r=0..N
  for c=rowS[r]..rowS[r+1] {
    res_i[r]+=data[c]*vec_i[cIdx[c]];
    check {fault<fail>(c > prevC);}
    prevC=c;
  }
 commit_CD(cd);
}
```

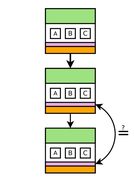


Generalization of Classical Resiliency Techniques

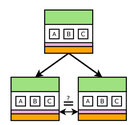
CDs represent a flexible **generalization of many prior techniques**, including:

1. Checkpoint-restart
   - central/distributed, coordinated/uncoordinated
2. Time and space-based redundancy (⟶)
3. Transactional programming models for reliability
   - Recovery Blocks, Argus, Relax, FaultTM

As such, CDs incorporate the benefits of existing techniques, and perform at least as well as traditional resiliency approaches.

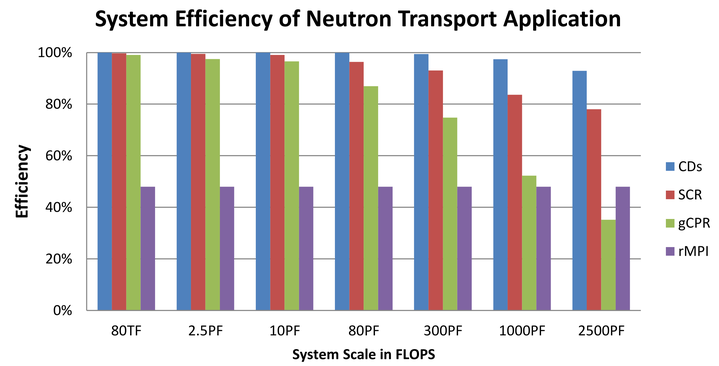


(c) Time



(d) Space

Preliminary Evaluation (Scalability)



CDs allow mapped applications to scale to larger system sizes or higher error rates.

Current Prototyping Efforts

Cray Inc. is developing a **CD research prototype** (**API and runtime**) targeting the **Cray XK6** (e.g., ORNL Titan). This prototype will explore the true overheads of flexible and hierarchical preservation and restoration.

Future Areas of Exploration

Ongoing and future work on CDs includes:

- **Minimal average-case re-execution** and **reduced data movement** can lead to **scalable energy savings**
- The ability to exploit **application-specific fault tolerance** and **rematerialization** to tradeoff computation for state preservation
- Operation under **GAS** and **shared memory** models
- **Cooperation** with **system checkpointers**
- Potential for **automatic optimization**
- Leveraging the rich domain-specific data latent in **DSLs**

This research was, in part, funded by the U.S. Government . The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

Mail: mbsullivan@utexas.edu

Web: http://lph.ece.utexas.edu/public/CDs (QR: ↑)