Building Replication Systems with PRACTI and PADRE
Discussing Two Papers

- PRACTI Replication
  N. Belaramani, M. Dahlin, L. Gao, A. Nayate, A. Venkataramani, P. Yalagandula, J. Zheng
  NSDI 2006

- PADRE: A Policy Architecture for Data Replication Systems
  N. Belaramani, M. Dahlin, A. Nayate, and J. Zheng
  Under Submission
Thank you to Nalini Belaramani for lending me her thesis proposal slides, from which I borrow liberally.
Introduction

Replication is used for data availability, fault-tolerance, performance, etc.

There are trade-offs for replication strategies: performance vs. consistency vs. availability vs. partition resilience...
Existing Systems are Complex

<table>
<thead>
<tr>
<th>System</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayou</td>
<td>20,425</td>
</tr>
<tr>
<td>Coda</td>
<td>167,564</td>
</tr>
<tr>
<td>Pangea</td>
<td>30,000</td>
</tr>
<tr>
<td>TierStore</td>
<td>22,566</td>
</tr>
<tr>
<td>TRIP</td>
<td>16,909</td>
</tr>
</tbody>
</table>
Can you make it easier to build new replication systems and extend existing ones?

Yes!

Separate policy from mechanism (PRACTI)

Policy is safety constraints + liveness strategy (PADRE)
Understanding the Design Space

- **Coda** - distributed file system that allows client side caching for disconnected operation
- **TRIP** - client-server multicast with a single writer, separate data and control channels
- **Bayou** - eventual consistency with pairwise server communication
- **Chain Replication** - linearly ordered servers, where head services query, tail services updates
- **Pangea** - object replication in a wide area file system, with peer-to-peer exchange of updates
- **TierStore** - object replication with hierarchical organization for disconnected operations
3 Properties of Replication Systems

- **Placement** - which nodes store copies of which data (PR)

- **Consistency** - which reads must see which writes (AC)

- **Topology** - client-server, hierarchical, ad hoc (TI)
No System Provides All 3

Partial Replication
TierStore

Coda
Chain Replication

Pangea

TRIP
Arbitrary Consistency

PRACTI
Bayou

Topological Independence
How to provide all 3 properties?

- Separate the control and data paths
- Causally ordered invalidation messages
- Unordered body messages
- Introduce imprecise invalidations
- Allow a single invalidation to summarize a set of invalidations
Every node has a **Core** to maintain local state.

The core exports a **Local API** for reading, writing, creating and deleting objects.

Modifications are appended to a **Log**, and then update the **Checkpoint**.

Nodes store an arbitrary subset of objects locally (i.e. PR).
For read misses, and to push updates, nodes use a log exchange protocol similar to Bayou, but differing in two ways:

- Bodies and invalidations are sent separately
- Imprecise invalidations summarize sets of precise invalidations, ensure no causal gaps in the log

Note that invalidations are causally ordered (allows AC despite PR and TI)
PRACTI

- The node’s **Controller** ensures the replication and topology policies

- Decides where to send invalidations, and which should be precise

- Selects from where to pre-fetch bodies and which bodies to fetch

- Selects which node should service a read miss
PRACTI API

13 Action

Subscriptions, fetches, connections, etc...

41 Events

Blocked requests, consistency change, message received, etc...
Building Systems with PRACTI

- Mechanism is right
  - Supports all three properties
- Interface is challenging
  - How do you specify the policies?
What are the policies?

- **Safety**
  - Consistency and durability are ensured by blocking requests until they do not violate certain invariants

- **Liveness**
  - View this as a routing decision
  - Where to go to satisfy a miss?
  - Where to send updates?
Safety Policy

- Add interception layers for read/write and update
- Define 5 points where a policy can provide a predicate and timeout
  - Blocks request until predicate is satisfied or timeout is reached
Safety Policy

5 points:

- ReadNowBlock, WriteBeforeBlock, ReadEndBlock, WriteEndBlock, ApplyUpdateBlock

Standard predicates:

- isValid(o), isCausal(o), haveBody(o), hasArrived(Nodes, p), maxStaleness(...)

Pre-build consistency libraries for: best effort, causal consistency, 1-copy serializability, open/close, sequential, linearizability

Allows for custom predicates
Liveness Policy

- Exposes 12 actions and 23 events that can be used for update and meta-data routing

- Example actions:
  - Add/remove push/pull inval subscription, fetch body, get current time...

- Example events:
  - Read blocked invalid object, write, inval arrives, fetch success...
Overlog/P2

Convenient language/runtime layer for specifying liveness policies.

Example

/* send ACK of the last received inval to predecessor */
tRdy04 sendAck(@Pred, Clk, Nodeld) :-
    readyToServe(@X, I), I==1,
    predecessor(@X, Pred), Pred != -1,
    latestInval(@X, Clk, Nodeld).
## List of Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Bayou*</th>
<th>CR#</th>
<th>Coda*</th>
<th>Pangaea</th>
<th>TierStore</th>
<th>TRIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong Consistency</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Weak Consistency</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>State Transfer</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Operation Transfer</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Structured Topology</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Ad-hoc Topology</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Callbacks</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Co-operative Caching</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Hoarding</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Anti-Entropy</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Flooding</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Safety</th>
<th>Liveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayou</td>
<td>20,425</td>
<td>12</td>
<td>29</td>
</tr>
<tr>
<td>CR</td>
<td>-</td>
<td>88</td>
<td>76</td>
</tr>
<tr>
<td>Coda</td>
<td>167,564</td>
<td>146</td>
<td>31</td>
</tr>
<tr>
<td>Pangea</td>
<td>30,000</td>
<td>12</td>
<td>67</td>
</tr>
<tr>
<td>TierStore</td>
<td>22,566</td>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td>TRIP</td>
<td>16,909</td>
<td>12</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>P-Code with liveness in P2</td>
<td>P-Code with liveness Java</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------</td>
<td>---------------------------</td>
<td></td>
</tr>
<tr>
<td>File open on cold cache</td>
<td>96.92 ms</td>
<td>10.44 ms</td>
<td></td>
</tr>
<tr>
<td>File open on hot cache</td>
<td>2.59 ms</td>
<td>0.79 ms</td>
<td></td>
</tr>
<tr>
<td>File close</td>
<td>112.1 ms</td>
<td>19.24 ms</td>
<td></td>
</tr>
</tbody>
</table>
The End