Motivation
- Key-value stores with few μs access times
  - RAMCloud, FaRM, MICA
- Everything in expensive DRAM
- Multi-tenancy needed to be practical
- And, get/put data models limit performance

Can we support safe user-level extensions?

Applications
- Push σ, π, γ
- Alternate data models, ADTs; graphs, TAO
- Heavy operations: matrices, GMM
- Our interest: pointer-chasing for concurrency control metadata

Requirements/Approaches

<table>
<thead>
<tr>
<th>Model</th>
<th>Fast Compile &amp; Install</th>
<th>Fast Runtime In/Out</th>
<th>Isolation</th>
<th>“Pointer Chasing”</th>
<th>Data-Intensive Ops</th>
<th>Compute-Intensive Ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓/✓</td>
<td>✓/✓</td>
<td>✓/✓</td>
</tr>
<tr>
<td>Native/C++</td>
<td>x</td>
<td>x</td>
<td>HW</td>
<td>x</td>
<td>Hard</td>
<td>✓</td>
</tr>
<tr>
<td>JavaScript</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓/✓</td>
<td>✓/✓</td>
<td>✓/✓</td>
</tr>
</tbody>
</table>

- (Near) native performance
- Low invocation overhead
- Runtime reconfigurable
- Inexpensive isolation
- Low installation overhead

Isolation Costs
- Data-intensive procedures mean pressure on protection domain boundaries
  Overhead: JS enter+exit 196 ns  JS to Native call+ret 31 ns

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Context Creation</th>
<th>Code Compilation</th>
<th>Context Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>V8::Context/JS</td>
<td>889 μs</td>
<td>427 μs</td>
<td>98 ns</td>
</tr>
<tr>
<td>Processes/C++</td>
<td>763 μs</td>
<td>58,000 μs</td>
<td>1.121 ns</td>
</tr>
<tr>
<td>VMFUNC/C++</td>
<td></td>
<td>58,000 μs</td>
<td>138 ns</td>
</tr>
<tr>
<td>stthreads/C++</td>
<td>2 μs</td>
<td>58,000 μs</td>
<td>150 ns</td>
</tr>
</tbody>
</table>

Performance Overheads

- Compared to native with no isolation
- JS 18-39% slower than native
- asm.js 2-10% slower than native

Procedure Dispatch
- Not designed for fast dispatch
- Single context can only admit one thread
- Cannot easily move contexts between cores
- Need smart management of context pool

Research Questions
- Right model for extending fast KVS?
- Other approaches
  - Rust type-safety
  - Native Client/SFI
- Dispatch
  - Contexts ↔ Cores
- Garbage collection
  - Regions?
  - Need to roll back globals too?
- Distributed Hotspot information
- JIT-performance vs Optimized C++