Reflections on Mobile Object Systems
Recent Reading …

• Explicit Code Mobility in Java RMI

Alexander Ahern and Nobuko Yoshida

Imperial College, London
Remote Invocation

```java
int m1(RemoteObject r, int a) {
    int x = r.f(a);
    int y = r.g(a, x);
    int z = r.h(a, y);
    return z;
}
```

• What’s wrong?
Ahern’s solution

Sender:

```java
int mOpt1(RemoteObject r, int a) {
    thunk<int> t = freeze(
        int x = r.f(a);
        int y = r.g(a, x);
        int z = r.h(a, y);
    );
    return r.run(t);
}
```

new method on r (at remote site):

```java
int run(thunk<int> x) {
    return defrost(x);
}
```
int m1(RemoteObject r, int a) {
    move this to r;
    int x = r.f(a);
    int y = r.g(a, x);
    int z = r.h(a, y);
    return z;
}
Distributed Shared Memory
The Basic Idea

- multiple processes share a single virtual memory space
- processes do loads/stores from/to memory locations
- pages may be resident (local) or non-resident (remote)
  - accesses to non-resident pages generate page faults
- page faults are handled by the OS and serviced by the DSM middleware
  - perhaps by retrieving the page from another machine
- protection faults can also be used by the DSM system to intercept “interesting” references to the shared memory
  - perhaps by invalidating pages on another machine
Characteristics

• Inter-process communication is via modification and subsequent reading of shared memory locations
  • Semantics defined by memory consistency model

• Local and remote communication look the same
  • remote communication is hidden behind MMU faults; invisible to the application
  • some memory accesses take (very much) longer than others
  • analogy with cache misses on SMPs

• Like programming a shared memory multiprocessor
  • UMA vs NUMA vs NORMA architectures
Key Issues and Challenges

1. Performance
2. Performance
3. Performance
Why is Performance Hard?

- minimum unit of communication is a page (why?)
- two processes accessing the same page cause thrashing
  - … even if they are accessing different addresses ("false sharing")
- effect of page size?
Replicated Pages

• Most DSM systems replicate pages
  • read-only processes can execute in parallel
  • only one replica can be writable
    • how do we choose it?
    • how do we invalidate/update outdated copies?
• Cost of DSM comes from protection faults and message exchanges
  • Strongly influenced by caching strategy and “memory consistency model”
Strong Memory Consistency

- Serializable (sequentially consistent) execution:
  - one copy semantics, axiom of assignment holds
- Linearizable — preserves real-time ordering too
How can we Implement Strong Consistency?

1. Centralized DSM
2. Migrating DSM
3. Read-only replication; central server for writes
4. Read-only replication; migrating server for writes
Did anyone ask the Programmer?

- Programmers who use shared memory use a high-level language with synchronization constructs!
- Arbitrary memory sharing will get you a C- even if your program “works”!
- Novel idea: ask the language implementor what abstractions help her to do her job!