HEY… You got your *Paradigm* in my *Operating System*!

Chris Matthews, Owen Stampflee, Yvonne Coady
University of Victoria

Jonathan Appavoo IBM Research

Marc E. Fiuczynski Princeton University

Robert Grimm New York University

*“People like to live in denial; thinking that programming shouldn't be *this* hard right? There must be an easier way, if only those pesky developers followed $fashionable_methodology_of_the_day...”*

-Pantelis Antoniou

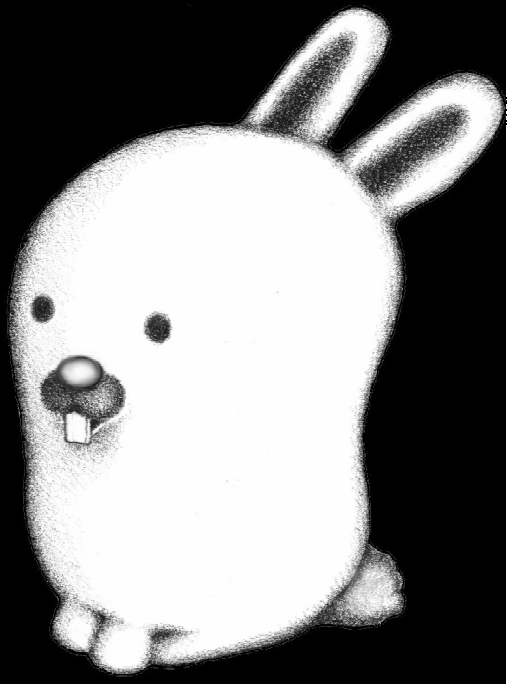
"*In fact, in Linux we did try C++ once already, back in 1992. It sucks. Trust me – writing kernel code in C++ is a BLOODY STUPID IDEA...*"

-Linus Torvalds

Should system developers buy-in to language mechanisms when they can selectively apply the paradigms for free?

Alef for Plan9

- The Alef language evolved from C, and was used in Bell Lab's Plan9 OS
- Alef sported:
  - Generics and regular ADTs
  - Error handling
  - Parallel task synchronization
  - Explicit tail recursion
  - Fat pointers



...it is clear that systems with multiple processors and multiple memory units are needed to provide greater capacity. This is not to say that fast processor units are undesirable, but that extreme system complexity to enhance this single parameter among many appears neither wise nor economic.

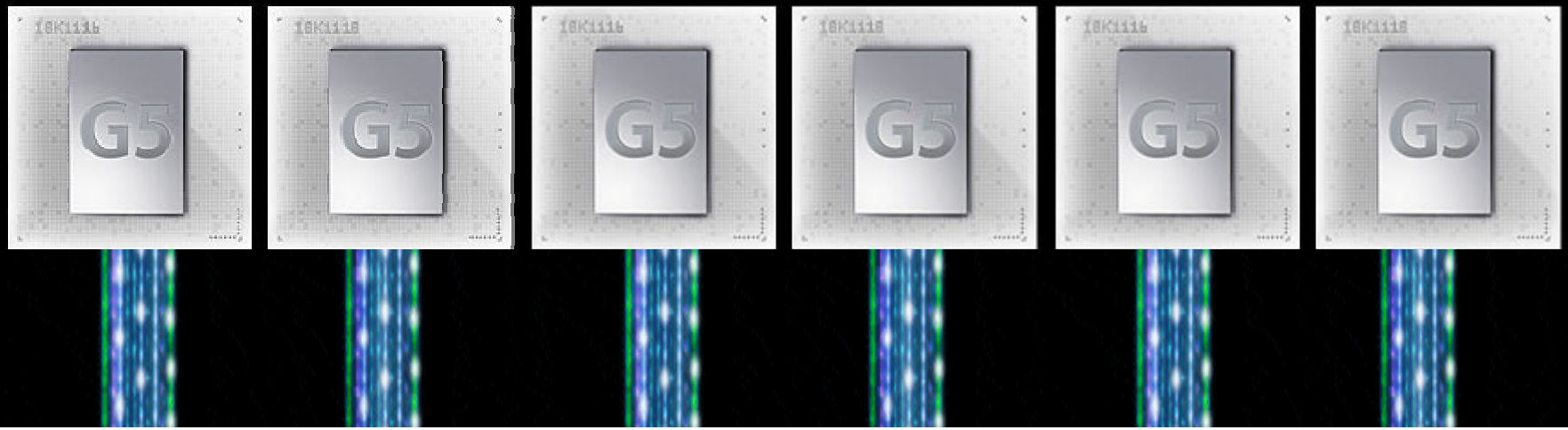
-Multics creators

Lots and lots of processors

- HEC is already using MP
  - IBM supercomputer: 65,536 700mhz ppc chips
- MP is becoming more common at home too.
- Both AMD and Intel are releasing dual core chips.

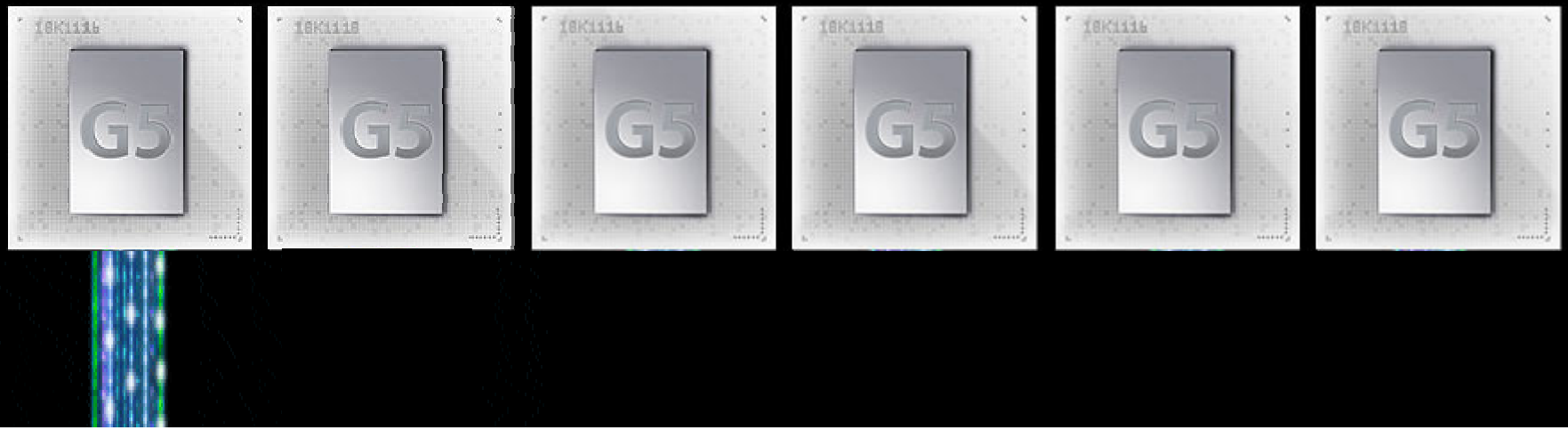
That pesky *"true"* concurrency

- On MP instructions are not interleaved, they are truly parallel
- Before disabling interrupts was all we needed to do before
- SMP allows the same natural programming model of uniprocessor



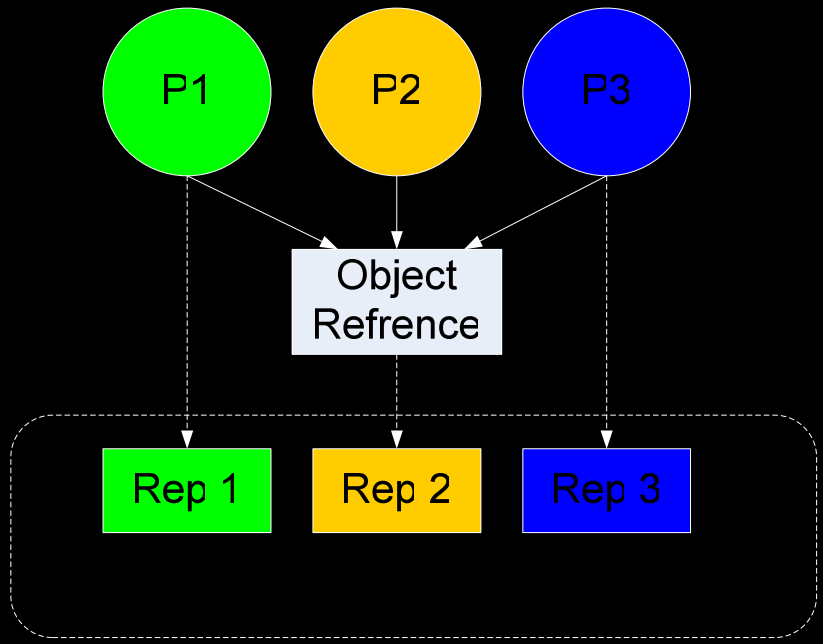
One to many

- OSes need to ensure mutual exclusion
- Put locks in the uniprocessor code path
- First ensure correctness, then incrementally remove the locks.
  - Forward progress?
  - Guarantees?
  - SMP locking, busy waiting? Cache?



Clustered Objects

- A simple model of partitioned objects to simplify the task of writing high performance SMP systems software
- Carefully constructed to take advantage of shared memory and reduce cache conflicts
- Already used in K42
- Why not in Linux?!?
- Where do COs belong?



Asking the right questions

- Clustered Objects make use of a C++ inheritance hierarchy
- Some things do not nicely fit the hierarchy though. Use AOP?
- Translation and Dispatch use assembly code and manipulate the programs vtable
- Was all this necessary? Did we really gain that much?

Managing Kernel Variants

- Process of creating a kernel variant:
  1. Download a mainline kernel from kernel.org
  2. Apply patches for specific domain
- Many of these patches represent cross cutting concerns
- Patch developers use simple text matching tools
  - Diff
  - Patch

An abstraction for crosscutting: Aspects

- We propose a semantic patching system for kernel code called C4
- In C4, aspects structure and modularize concerns that crosscut kernel code.

```
aspect {
  pointcut setuid() :
    execution(long sys_setreuid(..))  ||
    execution(long sys_setresuid(..)) ||
    execution(long sys_setuid(..));

  after setuid() { ckrm_cb_uid(); }

  pointcut setgid() :
    execution(long sys_setregid(..))  ||
    execution(long sys_setresgid(..)) ||
    execution(long sys_setgid(..));

  after setgid() { ckrm_cb_gid(); }
}
```

Should system developers buy-in to language mechanisms when they can selectively apply the paradigms for free?

Why Buy The Cow?
(when you can drink the milk for free)

- Traditional approaches are not sustainable
- An incremental approach that fits within current practices and political climate
- Through incremental adoption, we believe this approach will achieve buy-in from the systems community, otherwise skeptical of the *$fashionable_methodology_of_the_day*.
- We must think about problems not met by our current mechanisms, and what the important abstractions in the kernel really are.

- There is compelling evidence that with COs and crosscutting concerns as first class citizens in kernels, kernels would be better equipped to cope with software and hardware evolution.
- COs are implemented in C++, will this make it viable and interesting to the community?

*Should system developers buy-in to language mechanisms when they can **selectively** apply the paradigms for **free**?*

We say Yes!

But what are the barriers to acceptance?

Steps towards acceptance

- Accurate accounting of runtime and maintenance costs.
- Linguistic support
  - AOP?
  - IVY for metaprogramming?
- Refactoring for current code?
  - IVY macroscope?

C’est Fini

?