Device-driven I/O for Implicit Paging Operations

Jeremy Kerr <jk@ozlabs.org>

IBM Linux Technology Center

Outline

- K42 overview
- The K42 I/O system
- The device-driven model
- Implementation & experimentation
- Further work

K42 Overview

- Research OS
  - Intended for easy experimentation
- Microkernel-style architecture
- Fast user/kernel-space message passing system
  - Protected Procedure Call (PPC)
  - Stub compiler to easily define new calls
- Currently working on 64-bit POWER machines
- Compatible with Linux API & ABI

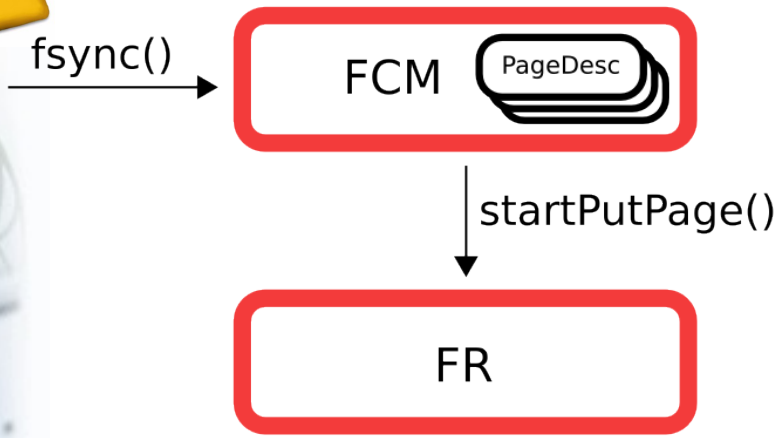
How it works now...

- K42 I/O subsystem
  - File objects are kept in userspace
  - write() is performed asynchronously
    - essentially a memcpy()
- Walk through a fsync...

Current fsync Path

File Cache Manager receives fsync()

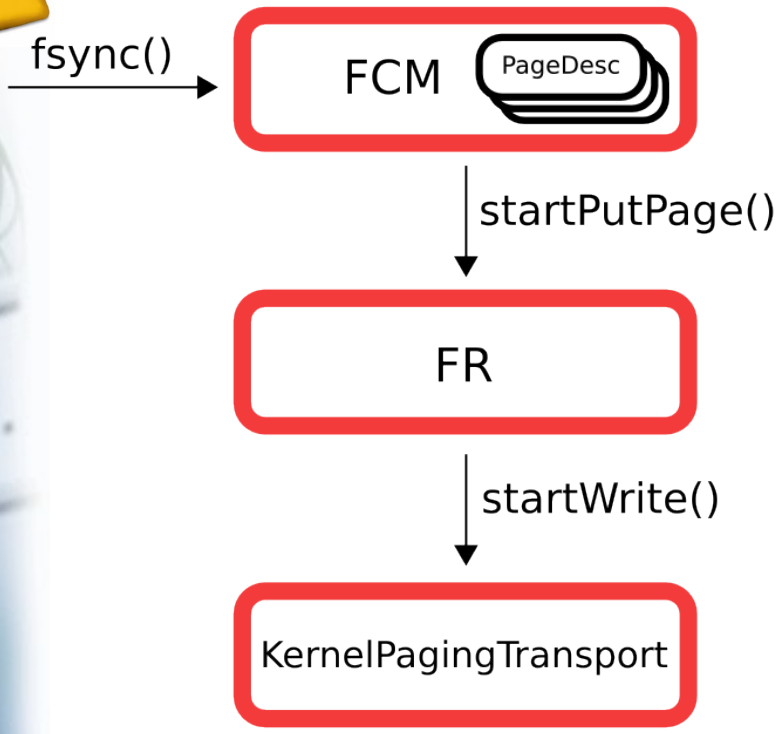
Current fsync Path



File Representative manages reference to userspace file

IBM

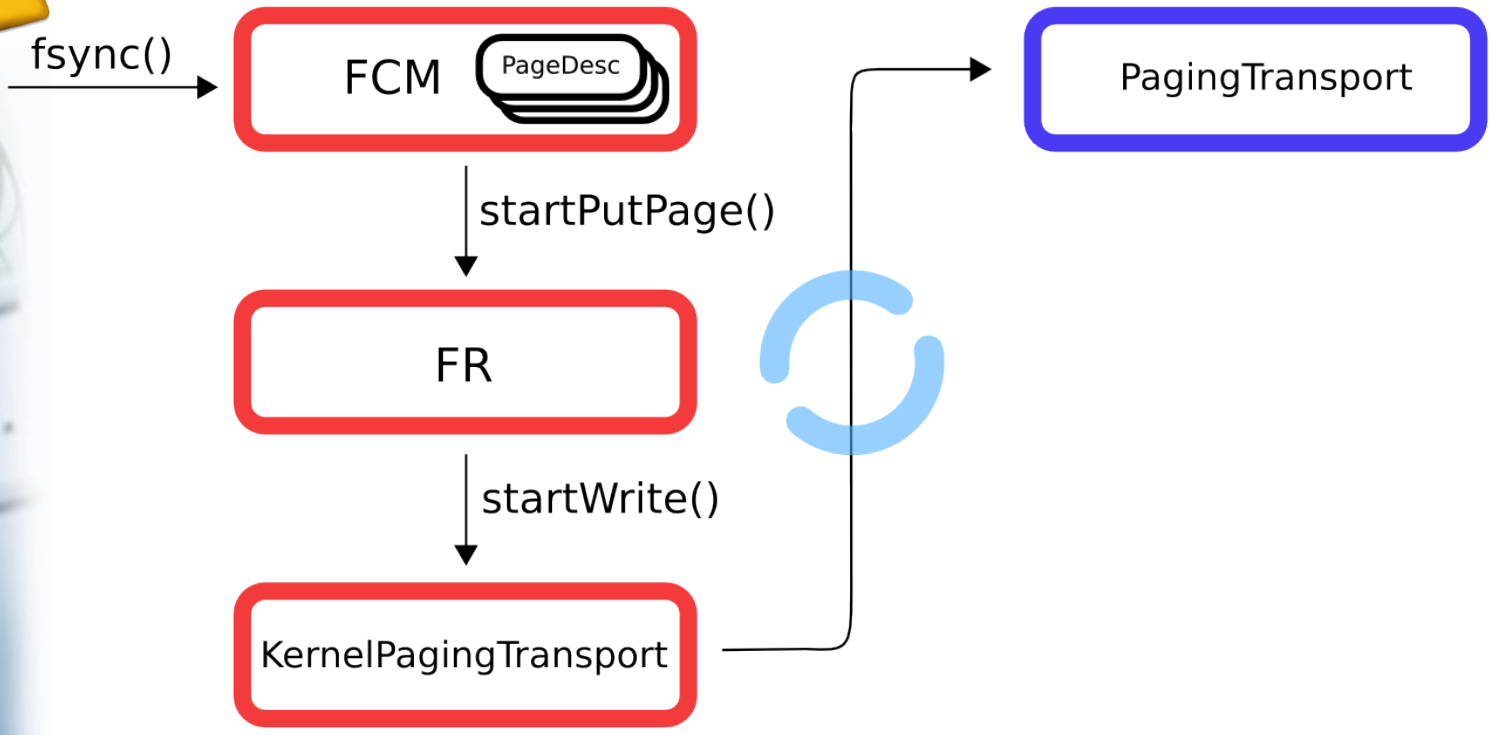
Current fsync Path



KernelPagingTransport passes I/O request to userspace...

IBM

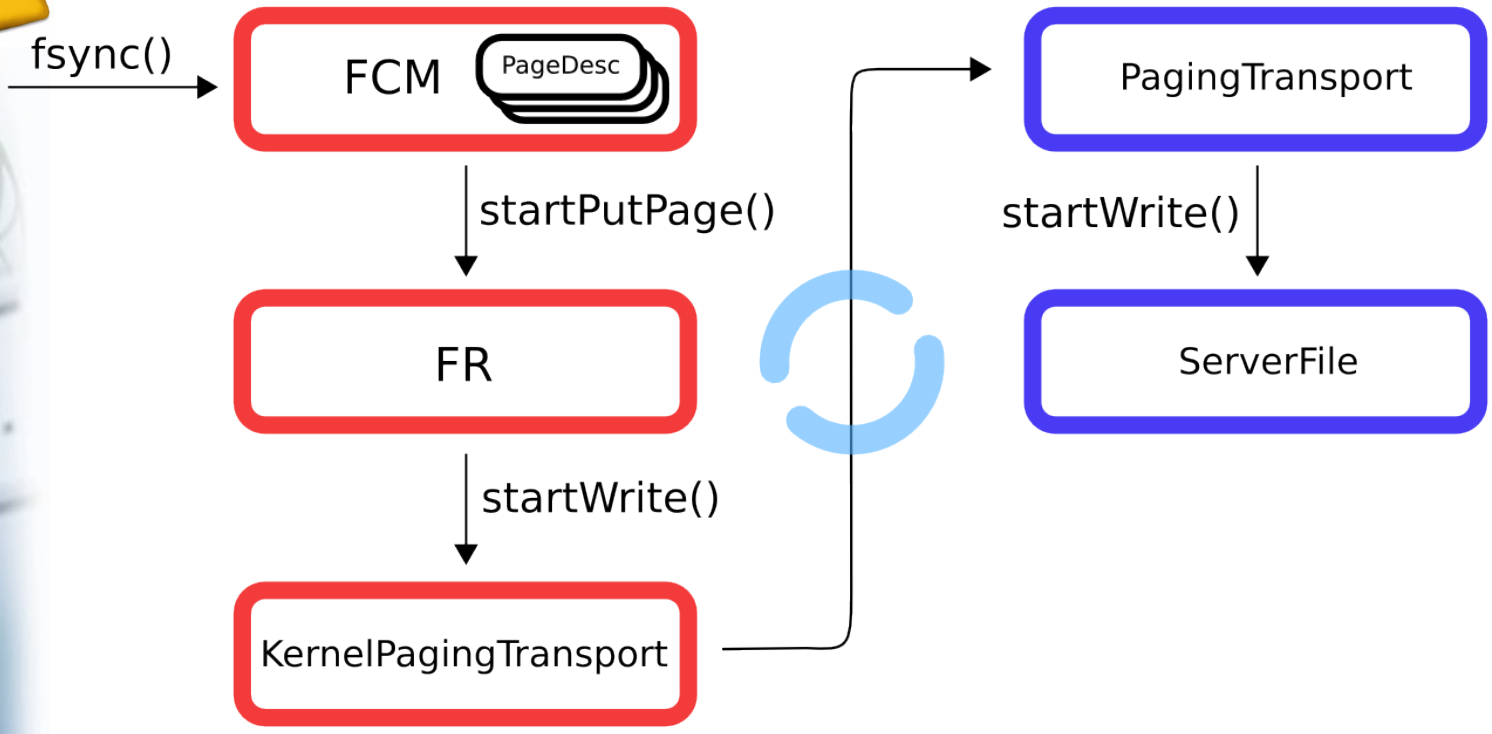
Current fsync Path



...over a shared ring-buffer, to the PagingTransport object

IBM

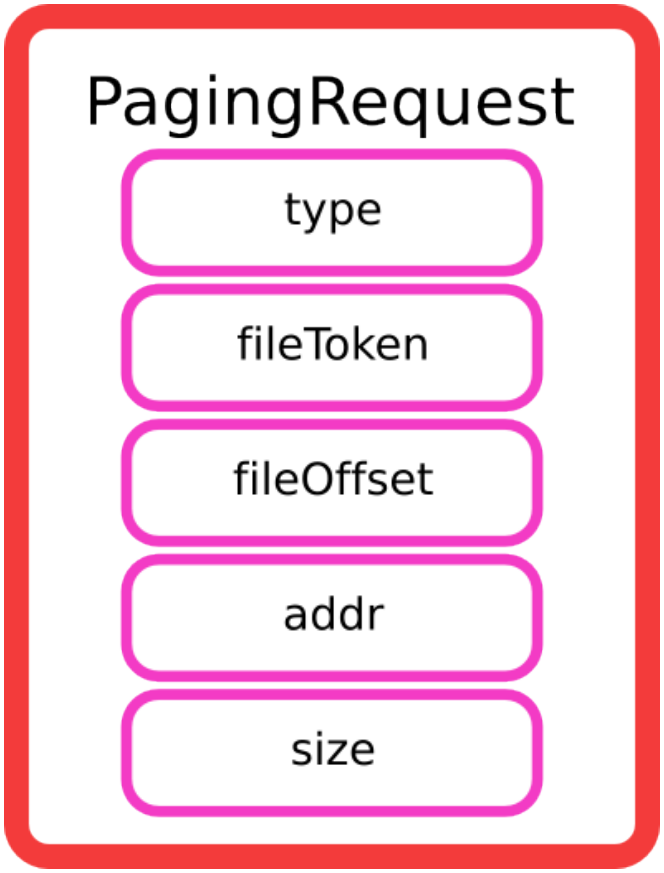
Current fsync Path



ServerFile object handles write operation

I/O Request Structure

- Basic unit of I/O
- Shared with userspace
- fileToken member
  - reference to ServerFile
  - opaque within kernel



IBM

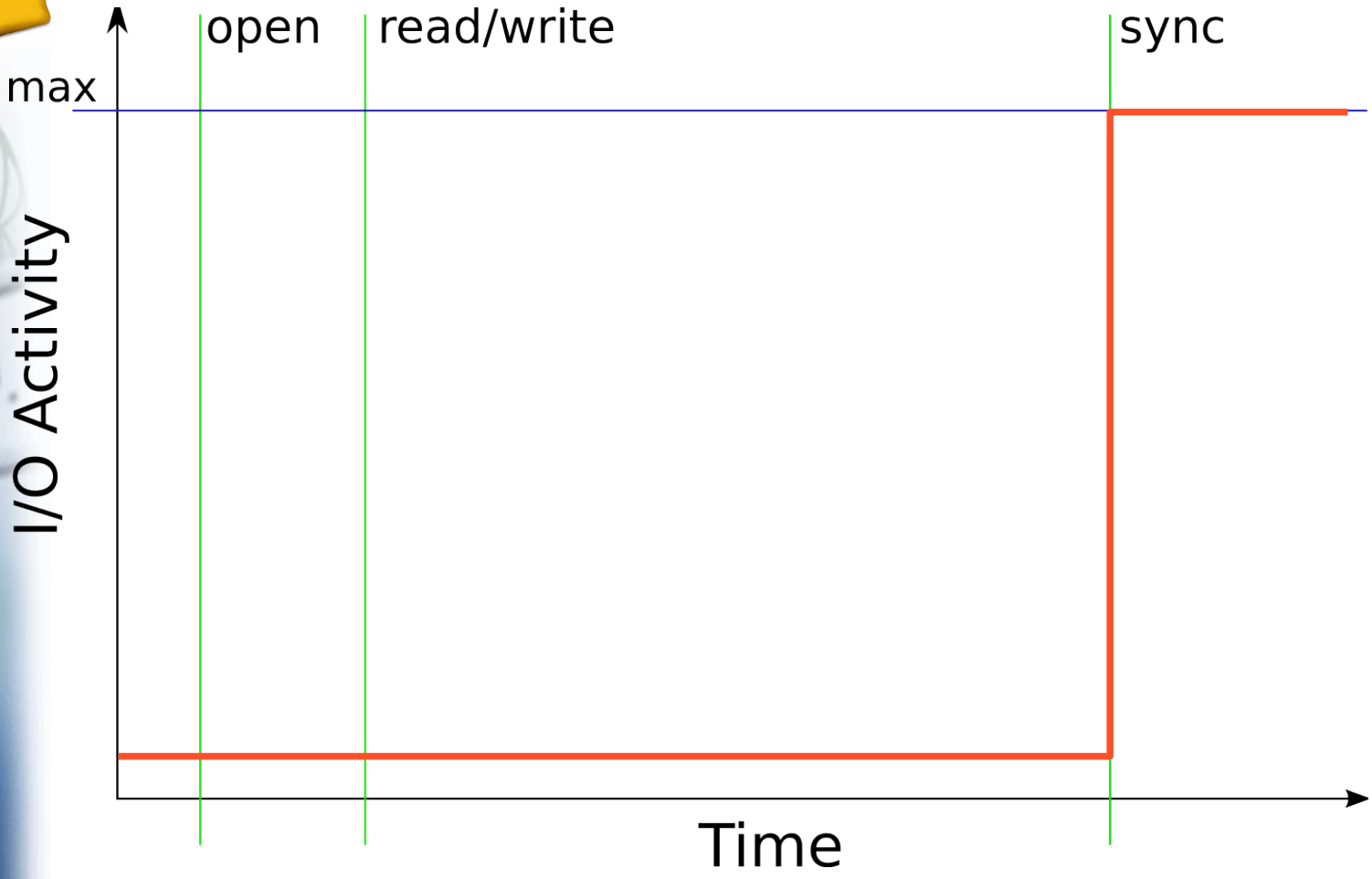
K42 I/O Subsystem

- In summary:
  - FCM (File Cache Manager)
    - Manages mapped regions
  - FR (File Representative)
    - Holds references to userspace file objects
  - ServerFile
    - Userspace file object
  - I/O performed by sending requests to userspace
    - Requests are “pushed” to the device

Device-driven I/O

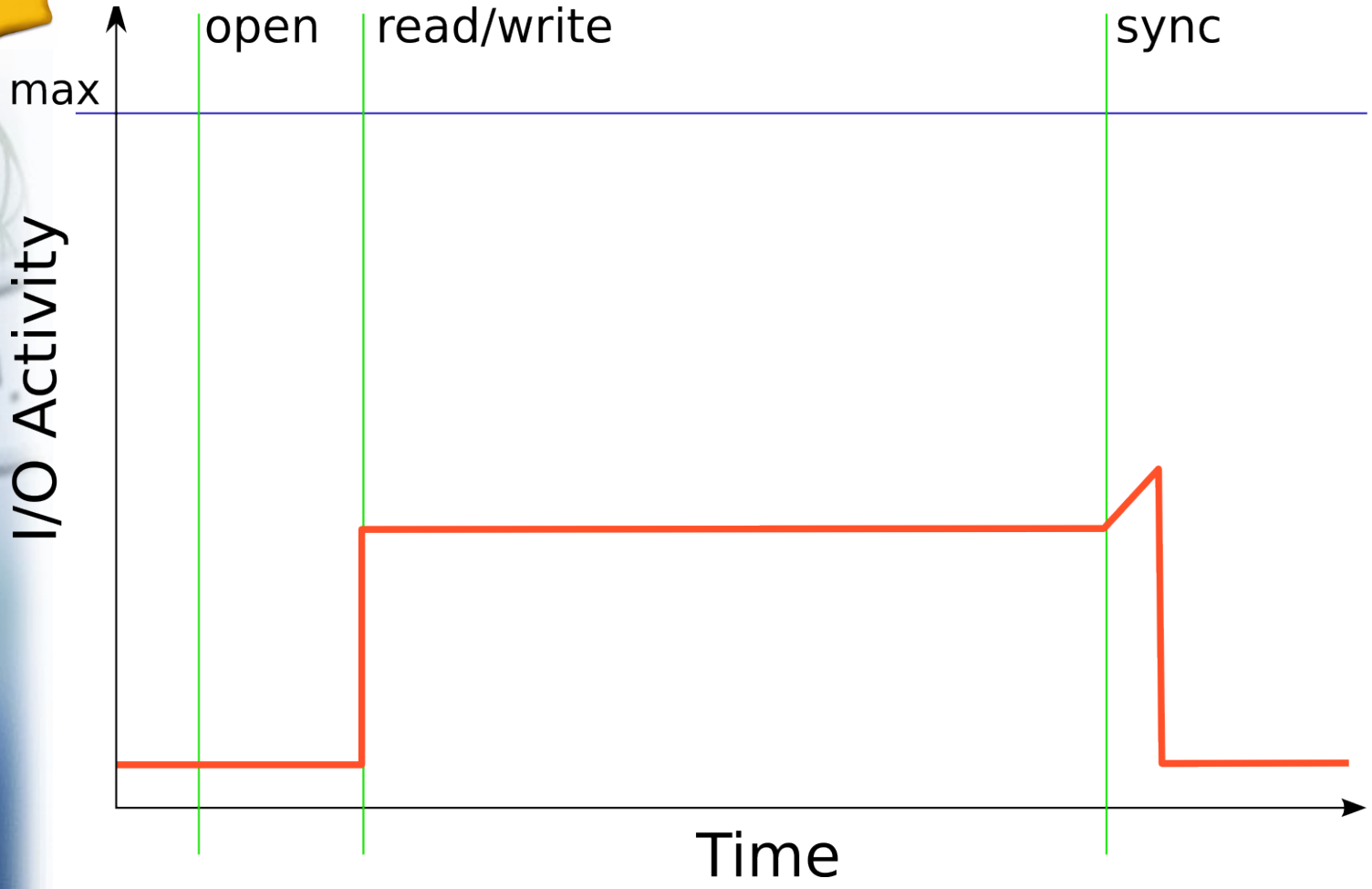
- Goal: Maximise usage of device bandwidth
  - Perform implicit operations when device is idle
- New programming model for I/O operations
  - FCMs no longer asynchronously generate implicit requests
  - Instead, FCMs are queried for requests
    - “pull model”

Current I/O Activity



IBM

Desired I/O Activity



IBM

Rationale

- Allows aggressive prefetch & page cleaning
  - Should stabilise bursty I/O
  - No need to be conservative
- FCMs “know best” how to optimise implicit ops
- Low request latency
  - Requests are only generated when the device is ready
  - No need for a cancellation mechanism

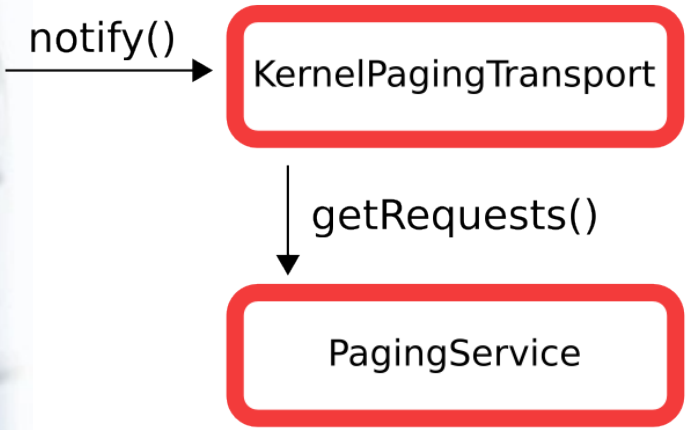
Design

- Keep track of pageable FCMs
  - New class: `PagingService`
- Generate requests
  - New method: `FCM::getRequests()`
- Notify the transport that bandwidth is available
  - New PPC: `KernelPagingTransport::notify()`

Device-driven Design

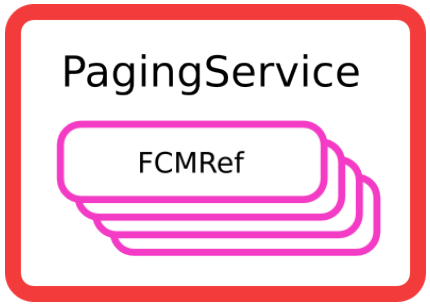
IBM

Device-driven Design



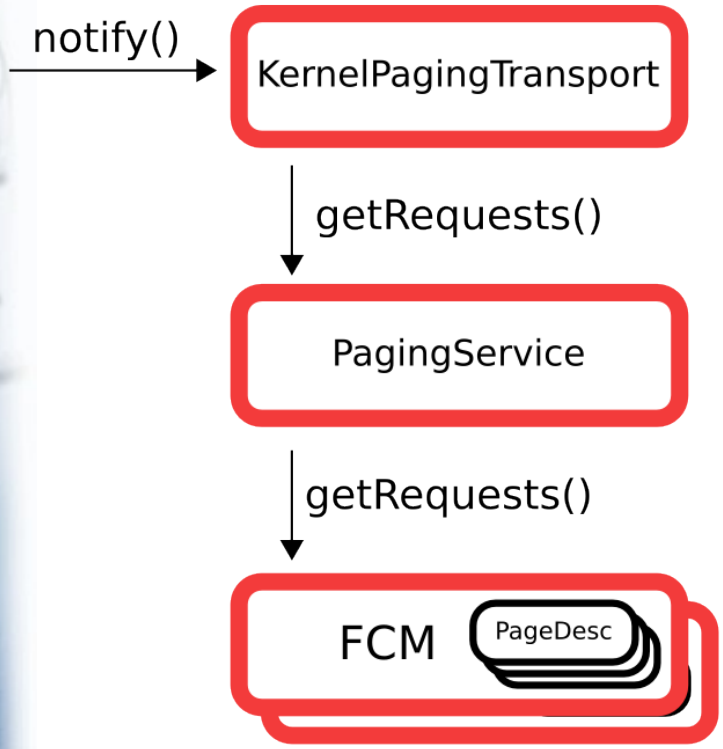
IBM

The PagingService



- Clustered object
  - One root object created at boot time
  - One “representative” created per processor
- FCMs register with the PagingService on creation
  - Only pageable FCMs
  - De-register on destruction
- Referenced by the KernelPagingTransport

Device-driven Design

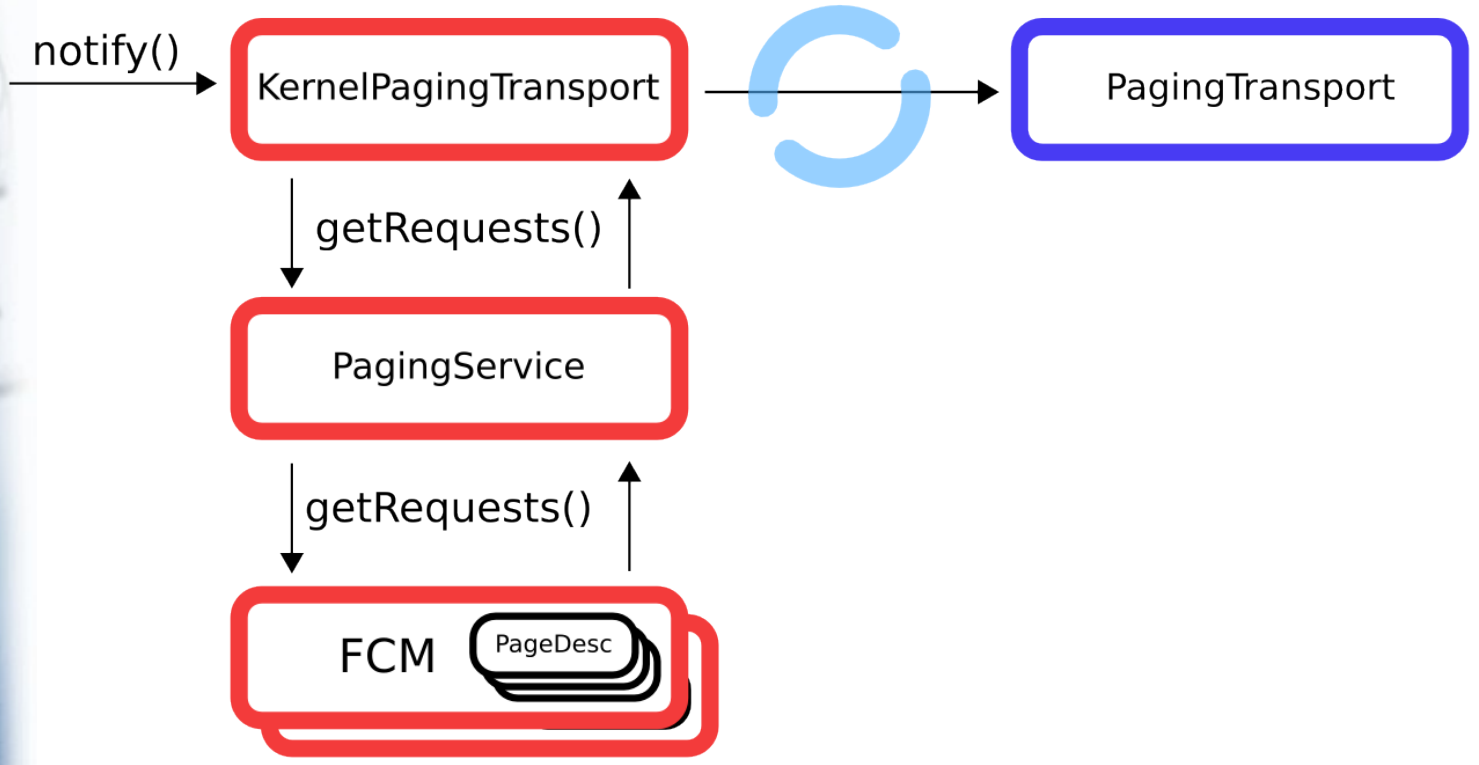


IBM

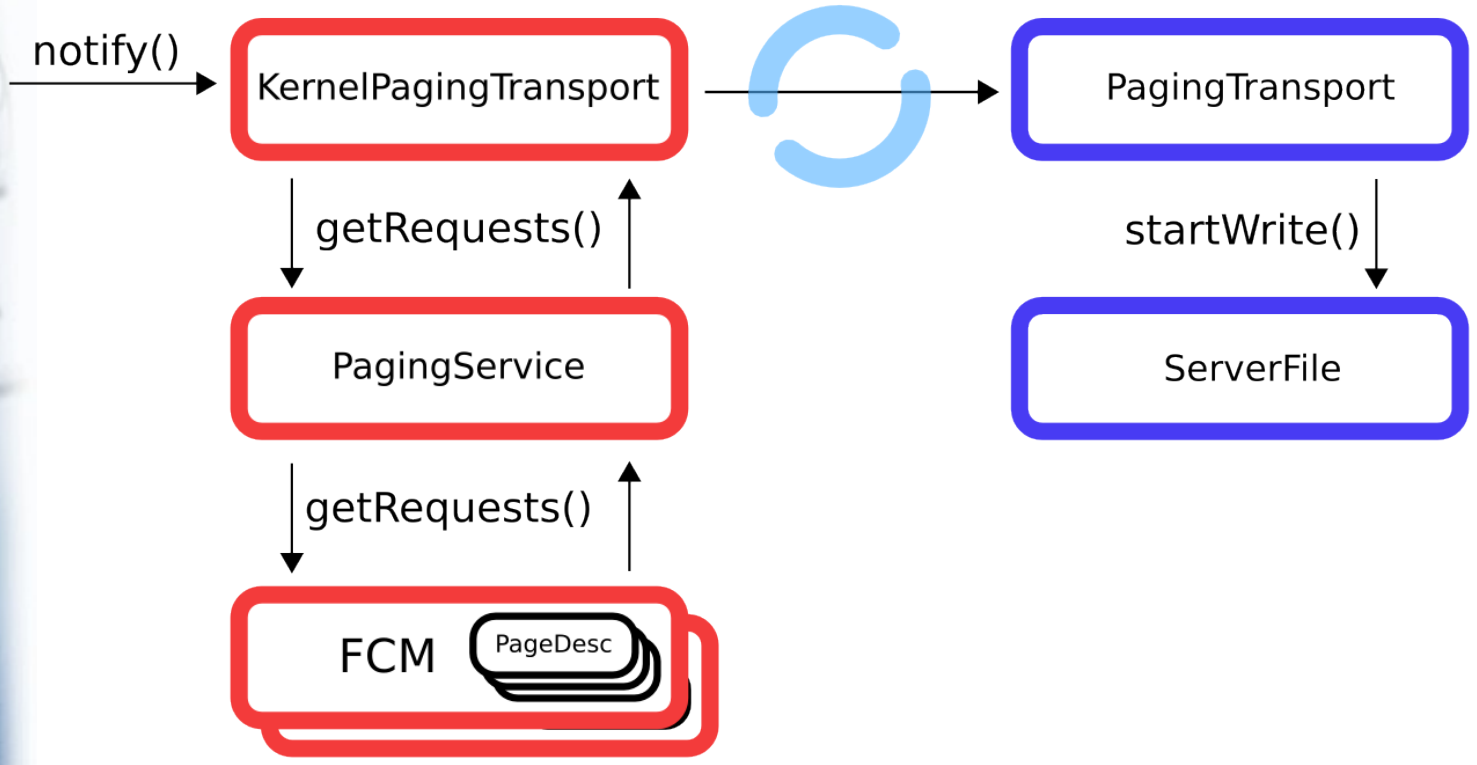
FCM::getRequests()

- Method to retrieve a set of I/O requests
  - Up to a specified maximum
- Implicit request pattern controlled by FCM
  - Can be based on history, expected patterns
    - Prefetch optimisation
    - Resident-set research at UofT
  - Easy to explore new ideas
    - FCM inheritance
  - Dynamic upgrade

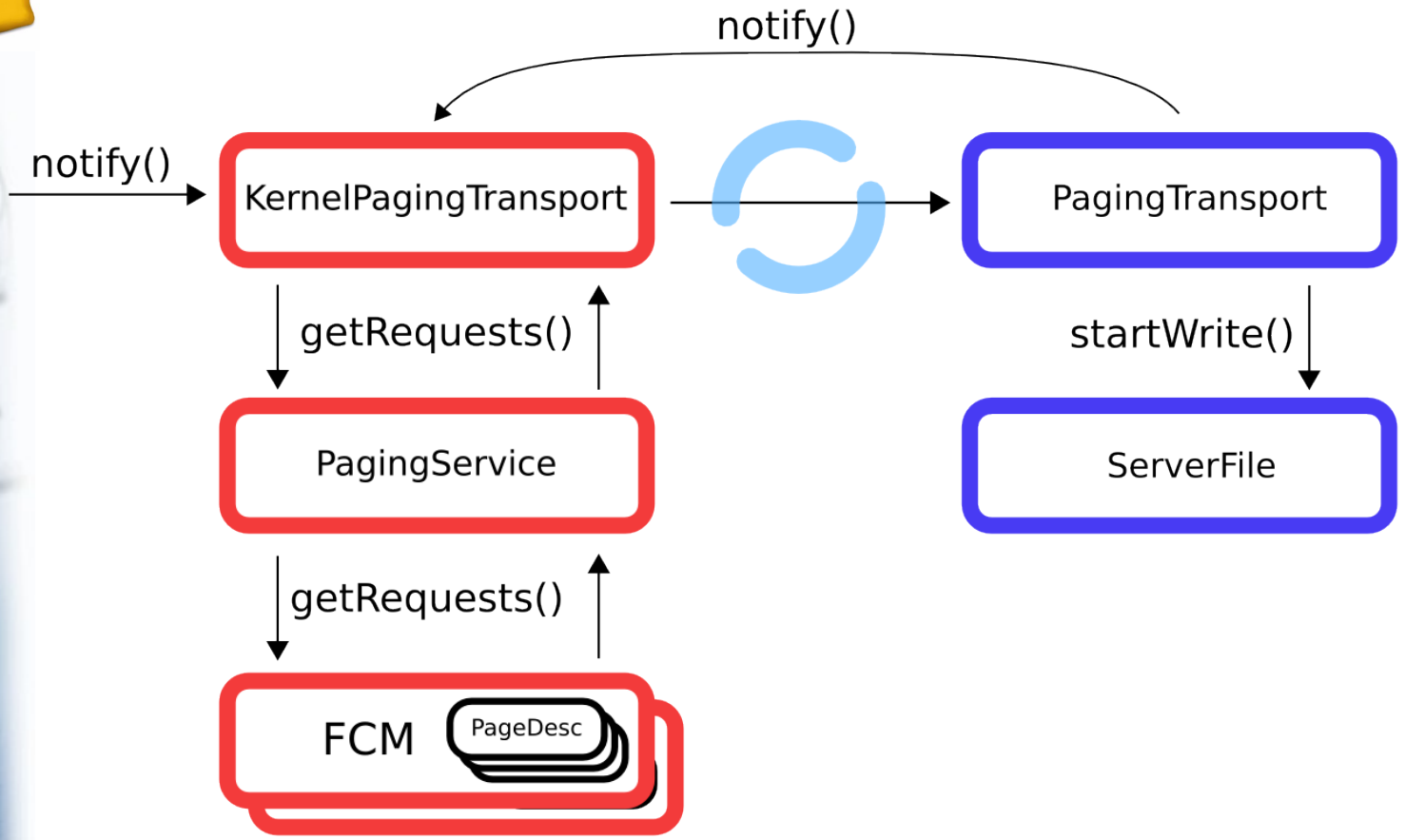
Device-driven Design



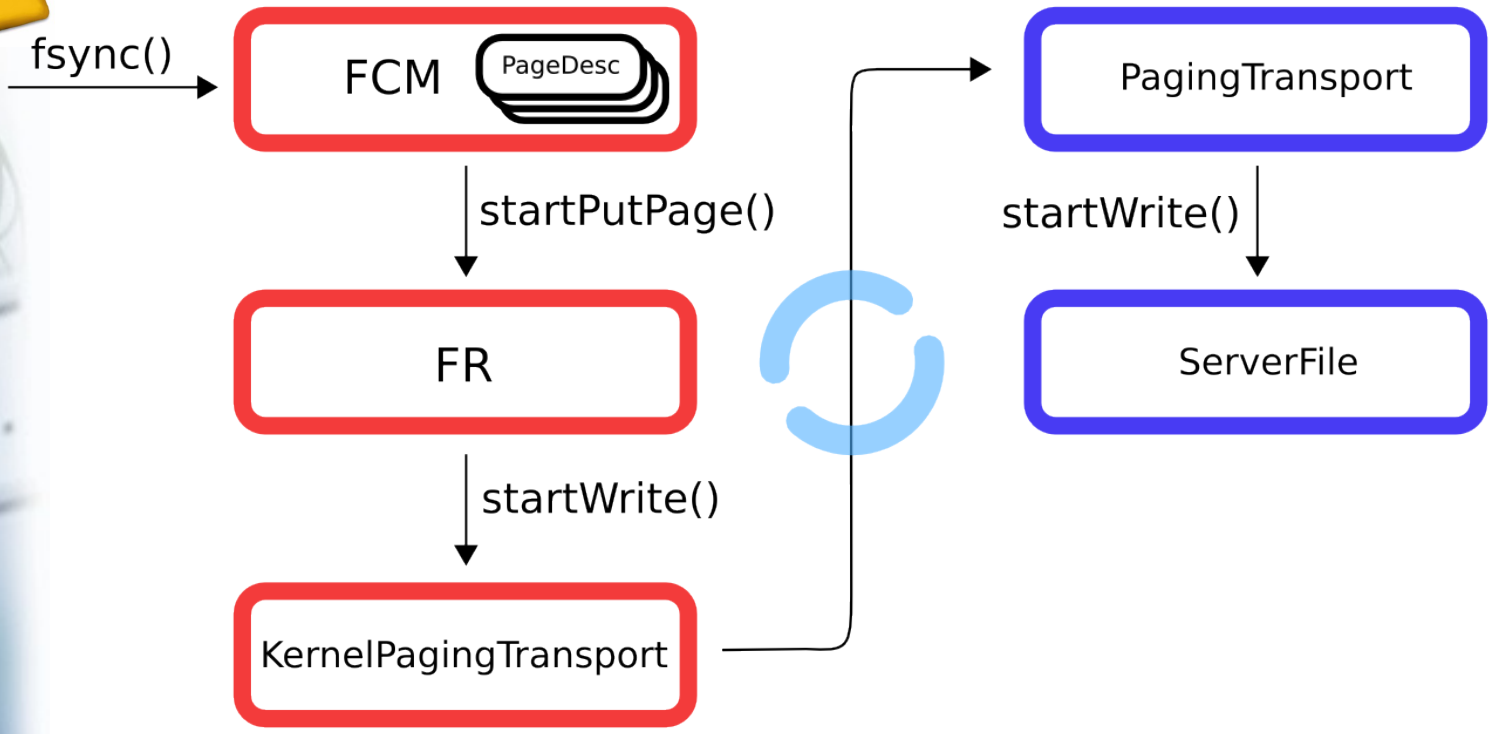
Device-driven Design



Device-driven Design

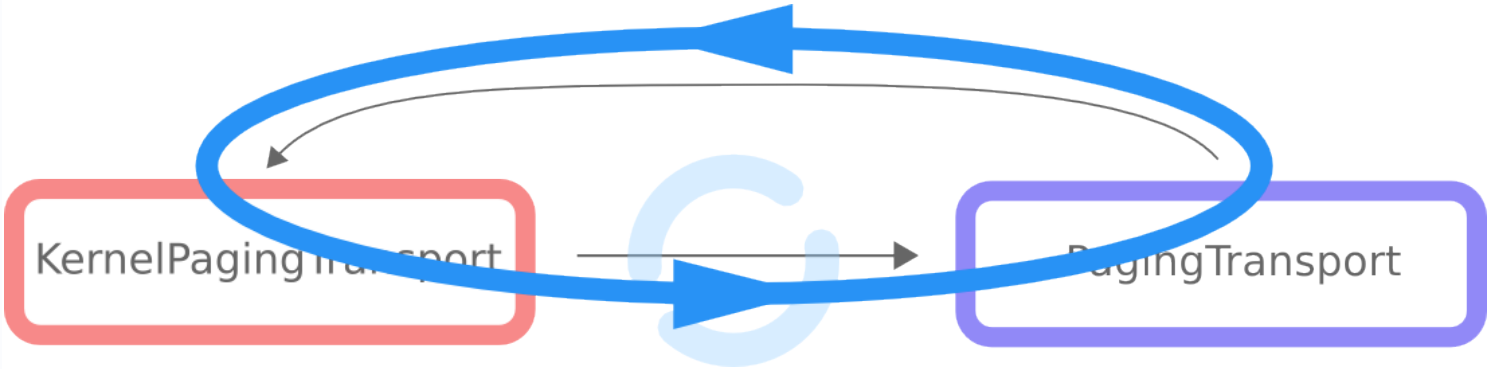


Design: Current Model



IBM

Flow Control



- Feedback loop provides flow control
  - Restricted by ring-buffer availability
  - Representative of device usage

IBM

Initial Concerns

- Global optimisation of implicit operations
  - No longer possible?
- Resource usage by PagingService
  - Don't want to tie up other resources
- Latency introduced for explicit operations

Tunables

- Implementation of FCM::getRequests()
  - Prefetch optimisation, page cleaning patterns
  - A whole area of research in itself...
- Allocation of bandwidth to FCMs
  - Which to query for requests?
  - Priority between FCMs

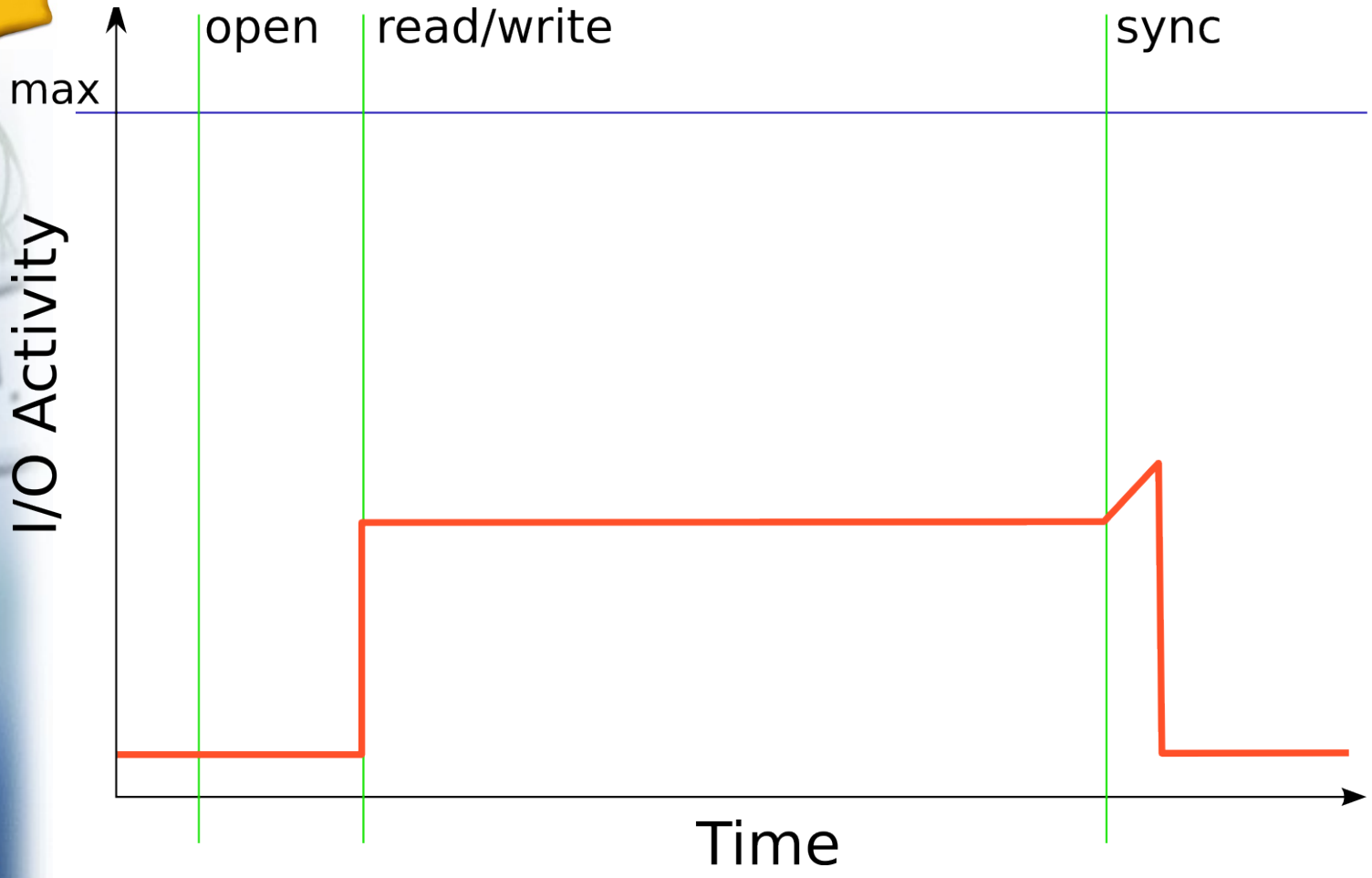
The Implementation

```
lib/libc/fslib/PagingTransport.C      |   13 +-
lib/libc/fslib/PagingTransport.H      |   11 +
lib/libc/io/FileLinux.C               |    8 -
os/kernel/ObjectRefsKern.H            |    4
os/kernel/bilge/CObjGlobalsKern.H     |    8 -
os/kernel/init/KernelInit.C           |    4
os/kernel/mem/FCM.C                   |    6 +
os/kernel/mem/FCM.H                   |    2
os/kernel/mem/FCMDefaultMultiRep.C    |    4
os/kernel/mem/FCMFile.C               |   55 ++++++++-
os/kernel/mem/FCMFile.H               |    6 +
os/kernel/mem/FR.H                    |   11 +
os/kernel/mem/FRCommon.H              |    6 +
os/kernel/mem/FRPA.C                  |   30 +++++
os/kernel/mem/FRPA.H                  |    3
os/kernel/mem/KernelPagingTransport.C |   60 +++++++++-
os/kernel/mem/KernelPagingTransport.H |   16 +-
os/kernel/mem/Makefile                |    7 -
os/kernel/mem/PagingService.C         |  204 ++++++++++++++++++++++++++++++++
os/kernel/mem/PagingService.H         |  166 ++++++++++++++++++++++++++
20 files changed, 585 insertions(+), 39 deletions(-)
```

Experimentation

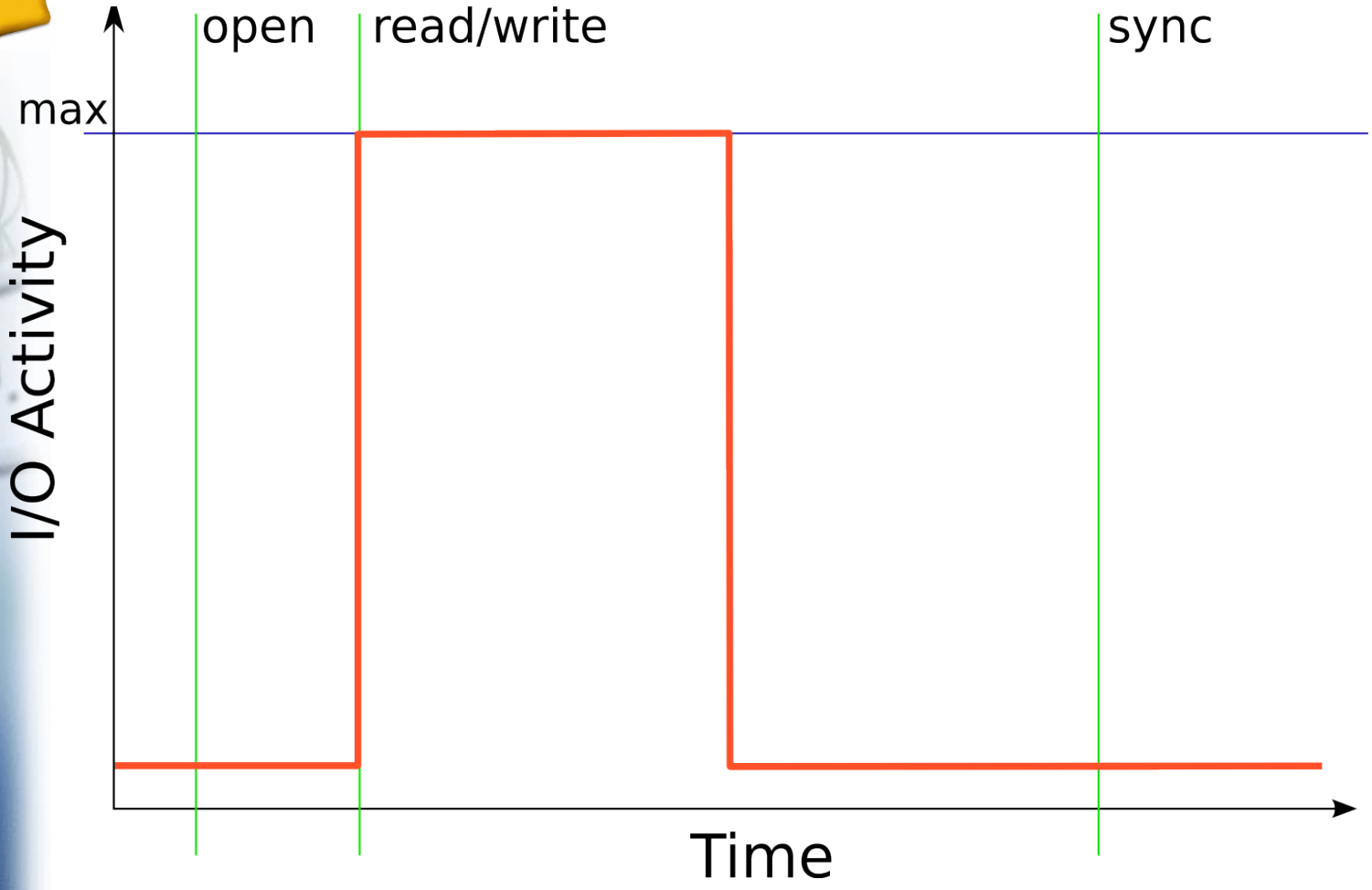
- Initial testing
  - Copy between filesystems
  - Time taken to sync changes to disk
- Implementing a prefetch oracle
  - Advance knowledge of pattern
- Alter prefetch oracle
  - Use history to determine pattern

Desired I/O Activity



IBM

Actual I/O Activity



IBM

Current Issues

- Current implementation is very aggressive
  - PagingService blocks other operations while paging
    - sync has no pages to flush
- High use of the PPC mechanism
  - Potential bottleneck

Further Work

- Tune scheduler to allow true background paging
- Improve FCM-request allocation
- Implementation of prefetch/cleaning algorithm
  - FCM::getRequests()

Resources

- K42 website at IBM Research
  - http://www.research.ibm.com/K42/
- Patches
  - http://ozlabs.org/~jk/projects/k42/

IBM

Legal

- This work represents the view of the authors and does not necessarily represent the view of IBM.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.