Preemptive Process Migration in a Cluster of Non-dedicated Workstations

Jan Čapek
jen@jikos.cz

Department of Computer Science and Engineering
Czech Technical University in Prague

23.6.2005

Outline

Outline

Clondike Cluster Layout

CCN

Basic Terminology

CCN Cluster Core Node - dedicated cluster node.

CDN Cluster Detached Node - non-dedicated, typically regular workstations.

CLoNDIKe CLuster Of Non-Dedicated Inter-operating Kernels.

Clondike Cluster Layout

CCN

CDN

CDN

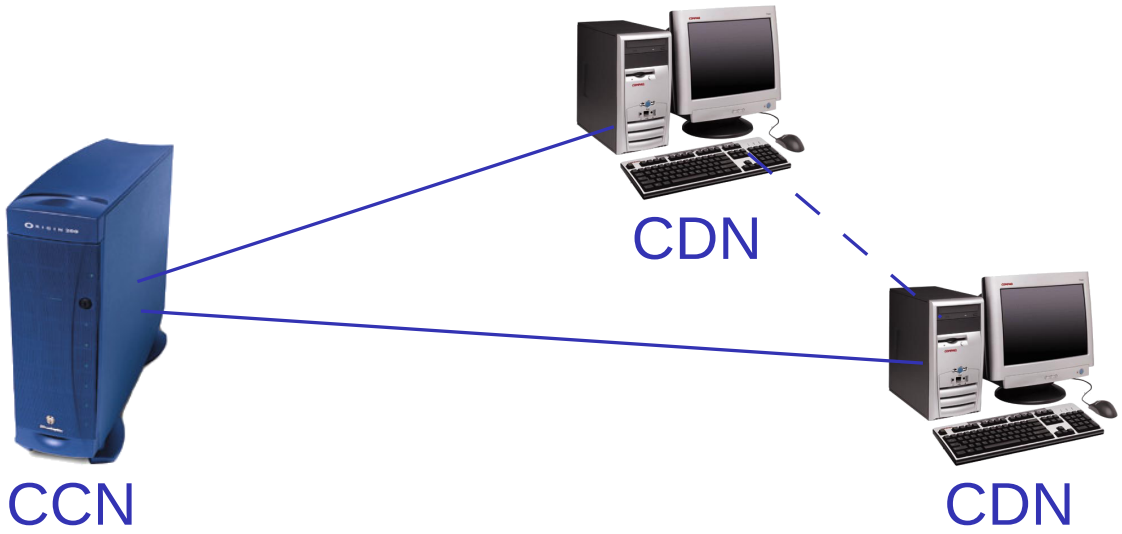
Basic Terminology

CCN Cluster Core Node - dedicated cluster node.

CDN Cluster Detached Node - non-dedicated, typically regular workstations.

CLoNDIKe CLuster Of Non-Dedicated Inter-operating Kernels.

Clondike Cluster Layout



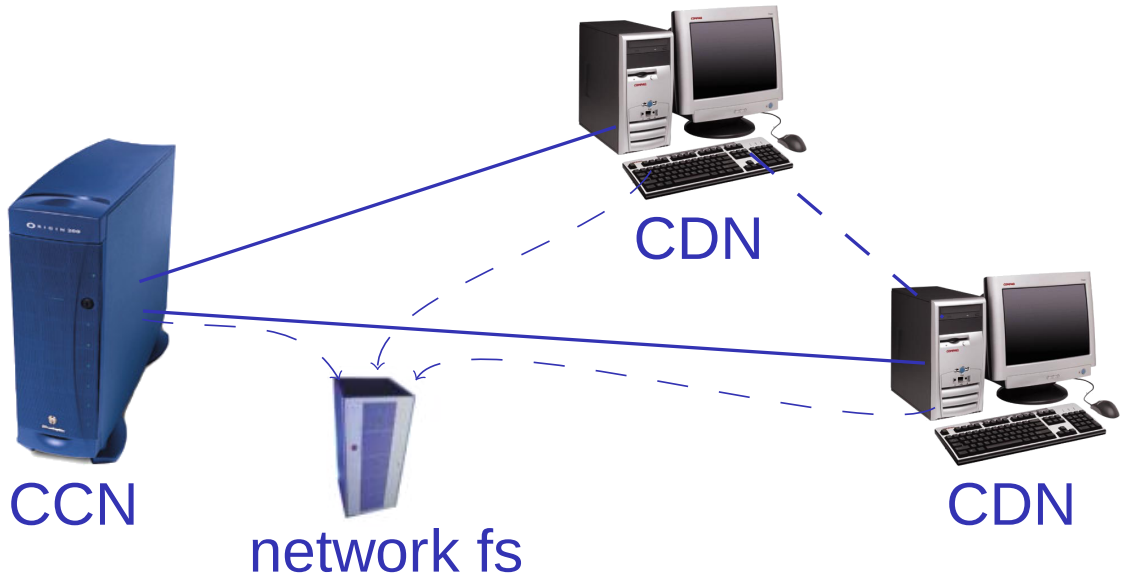
Basic Terminology

CCN Cluster Core Node - dedicated cluster node.

CDN Cluster Detached Node - non-dedicated, typically regular workstations.

CLoNDIKe CLuster Of Non-Dedicated Inter-operating Kernels.

Clondike Cluster Layout



Basic Terminology

CCN Cluster Core Node - dedicated cluster node.

CDN Cluster Detached Node - non-dedicated, typically regular workstations.

CLoNDIKe CLuster Of Non-Dedicated Inter-operating Kernels.

Process Migration

Migration Terminology

process Entity executing a program

- **PEN** Process Execution Node
- **UHN** Unique Home Node, origin of a migrating process.
- **PPM** Preemptive Process Migration, PEN change at any instant moment
- **NPM** Non-preemptive Process Migration, PEN change at specific moments only.

- Constraint: processes originate from CCN's only.
- Initially for a process in cluster: CCN = UHN = PEN

Process Migration

Migration Terminology

process Entity executing a program

PEN Process Execution Node

UHN Unique Home Node, origin of a migrating process.

PPM Preemptive Process Migration, PEN change at any instant moment

NPM Non-preemptive Process Migration, PEN change at specific moments only.

- Constraint: processes originate from CCN's only.
- Initially for a process in cluster: CCN = UHN = PEN

Process Migration

Migration Terminology

process Entity executing a program

PEN Process Execution Node

UHN Unique Home Node, origin of a migrating process.

PPM Preemptive Process Migration, PEN change at any instant moment

NPM Non-preemptive Process Migration, PEN change at specific moments only.

- Constraint: processes originate from CCN's only.
- Initially for a process in cluster: CCN = UHN = PEN

Process Migration

Migration Terminology

process Entity executing a program

PEN Process Execution Node

UHN Unique Home Node, origin of a migrating process.

PPM Preemptive Process Migration, PEN change at any instant moment

NPM Non-preemptive Process Migration, PEN change at specific moments only.

- Constraint: processes originate from CCN's only.
- Initially for a process in cluster: CCN = UHN = PEN

Process Migration

Migration Terminology

process Entity executing a program

PEN Process Execution Node

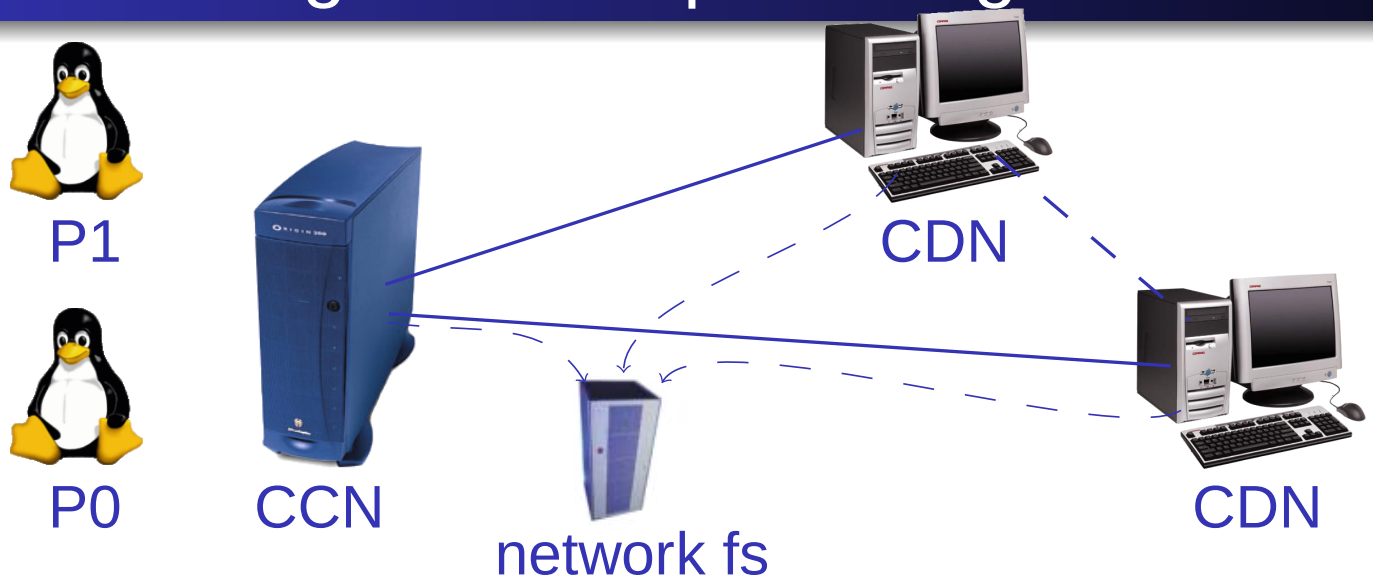
UHN Unique Home Node, origin of a migrating process.

PPM Preemptive Process Migration, PEN change at any instant moment

NPM Non-preemptive Process Migration, PEN change at specific moments only.

- Constraint: processes originate from CCN's only.
- Initially for a process in cluster: CCN = UHN = PEN

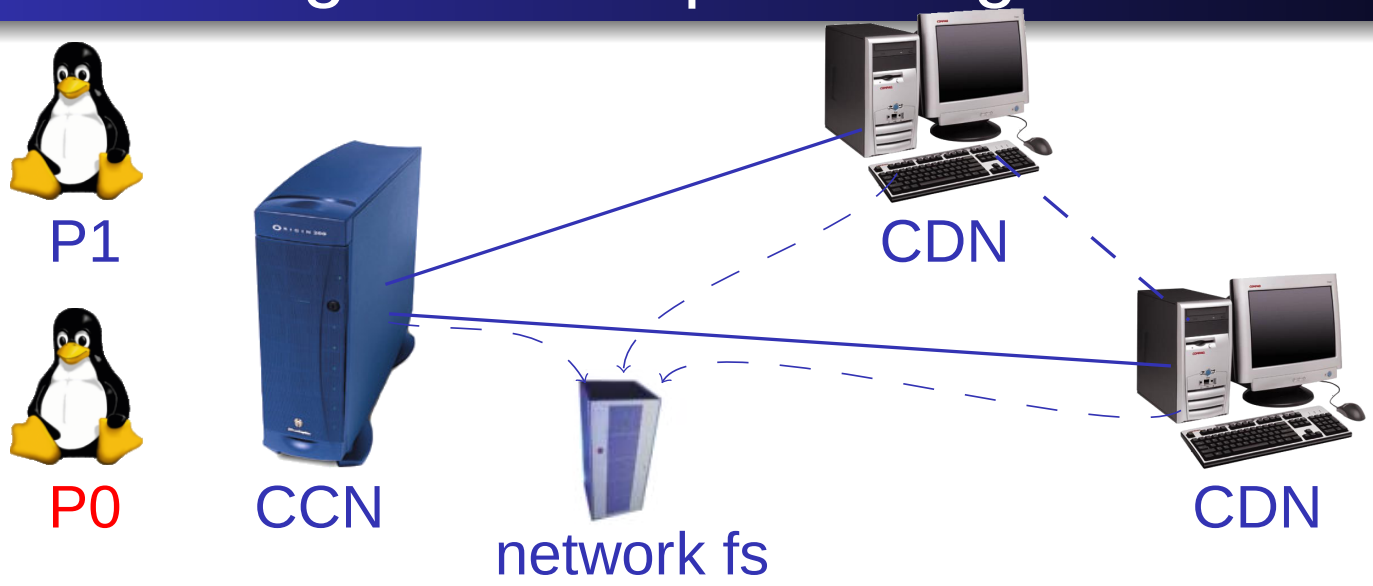
Process Migration Steps - Emigration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_0)$.

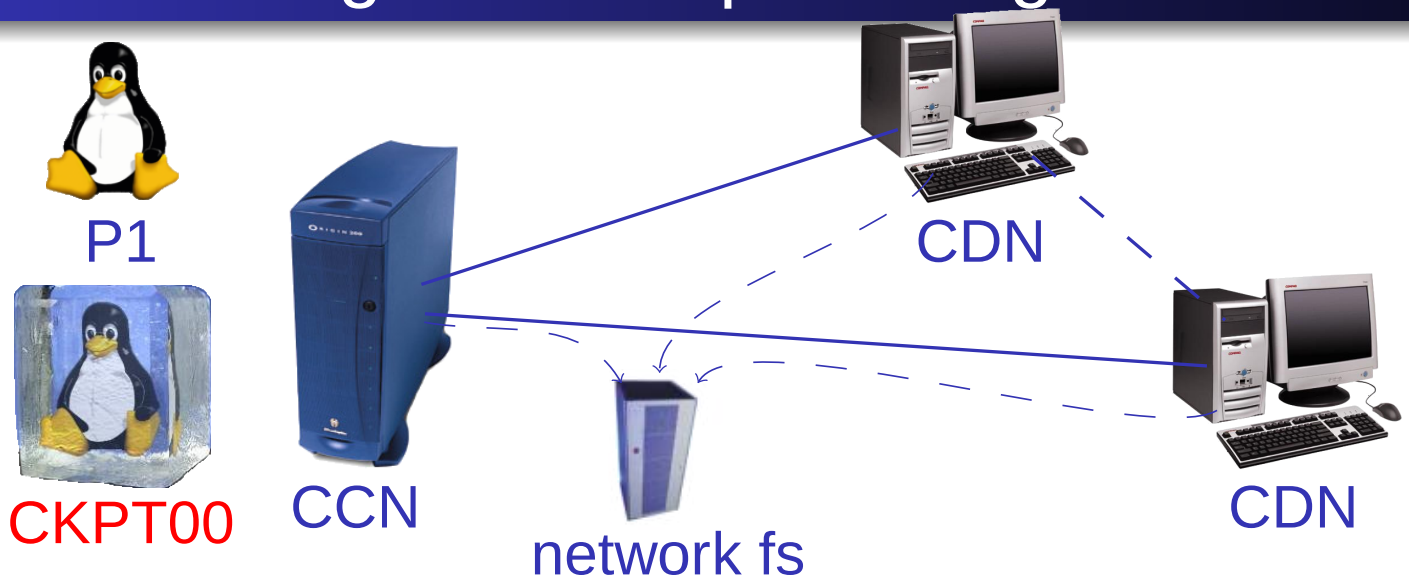
Process Migration Steps - Emigration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_0)$.

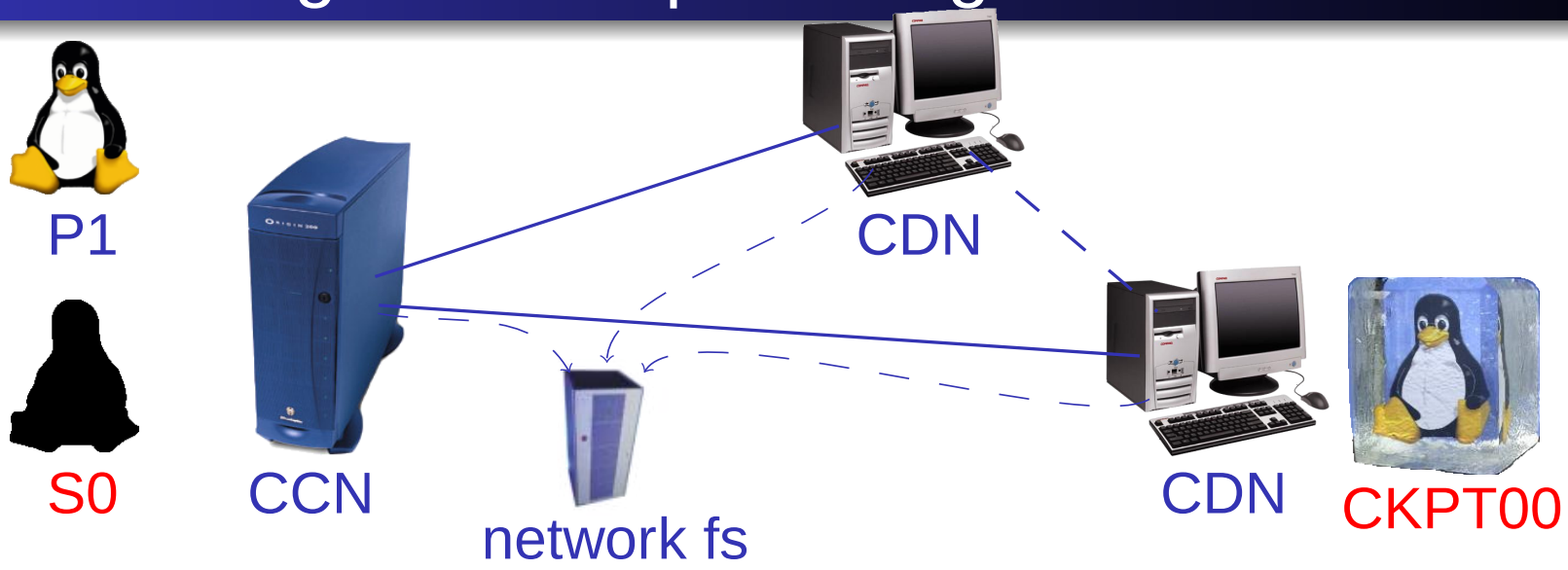
Process Migration Steps - Emigration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_0)$.

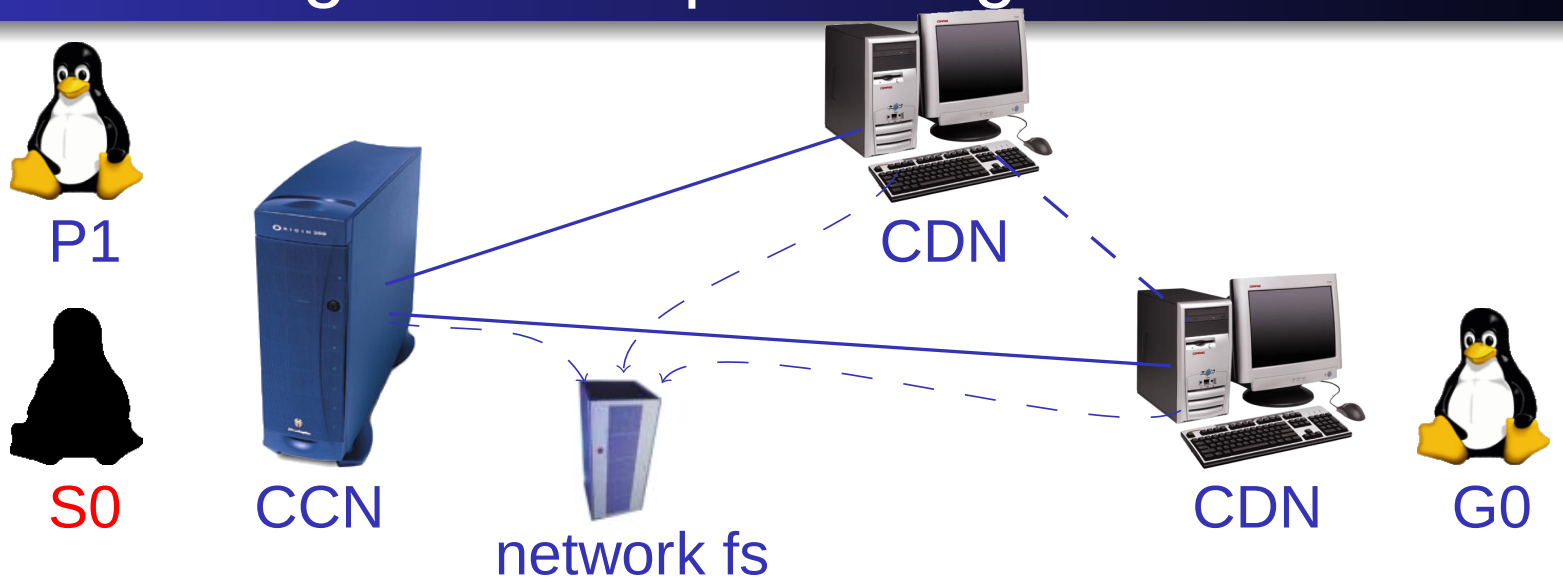
Process Migration Steps - Emigration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_0)$.

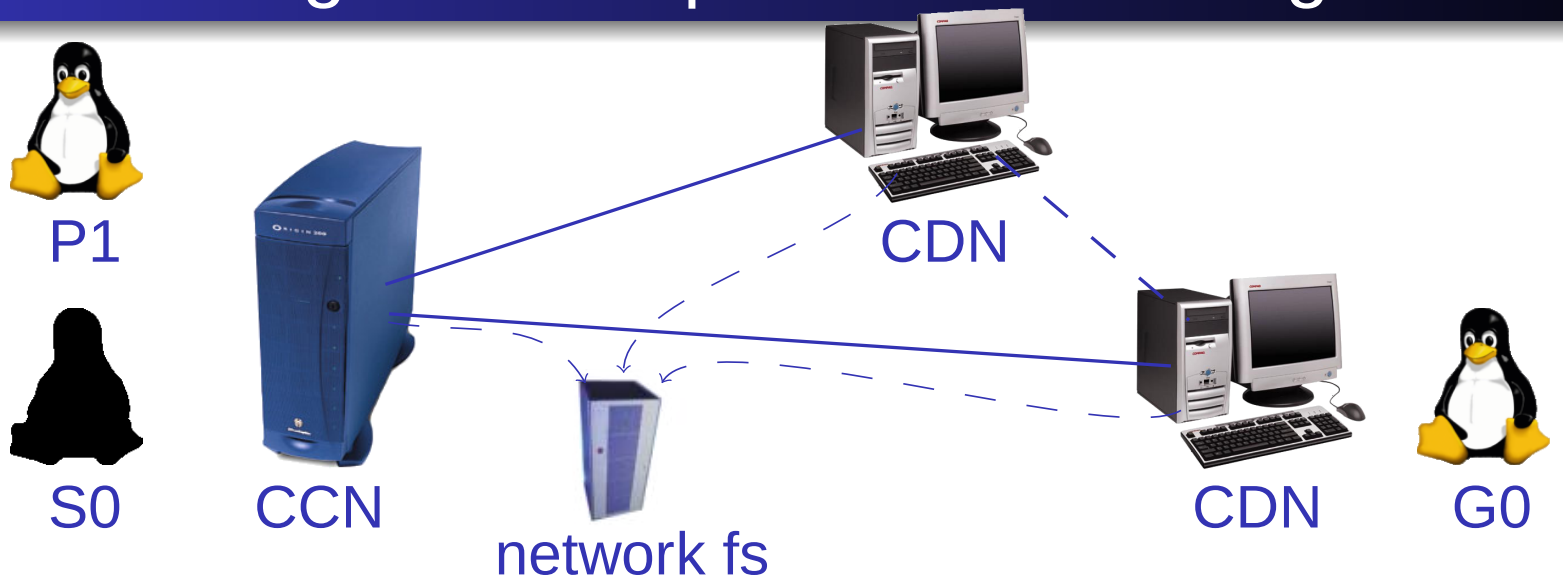
Process Migration Steps - Emigration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_0)$.

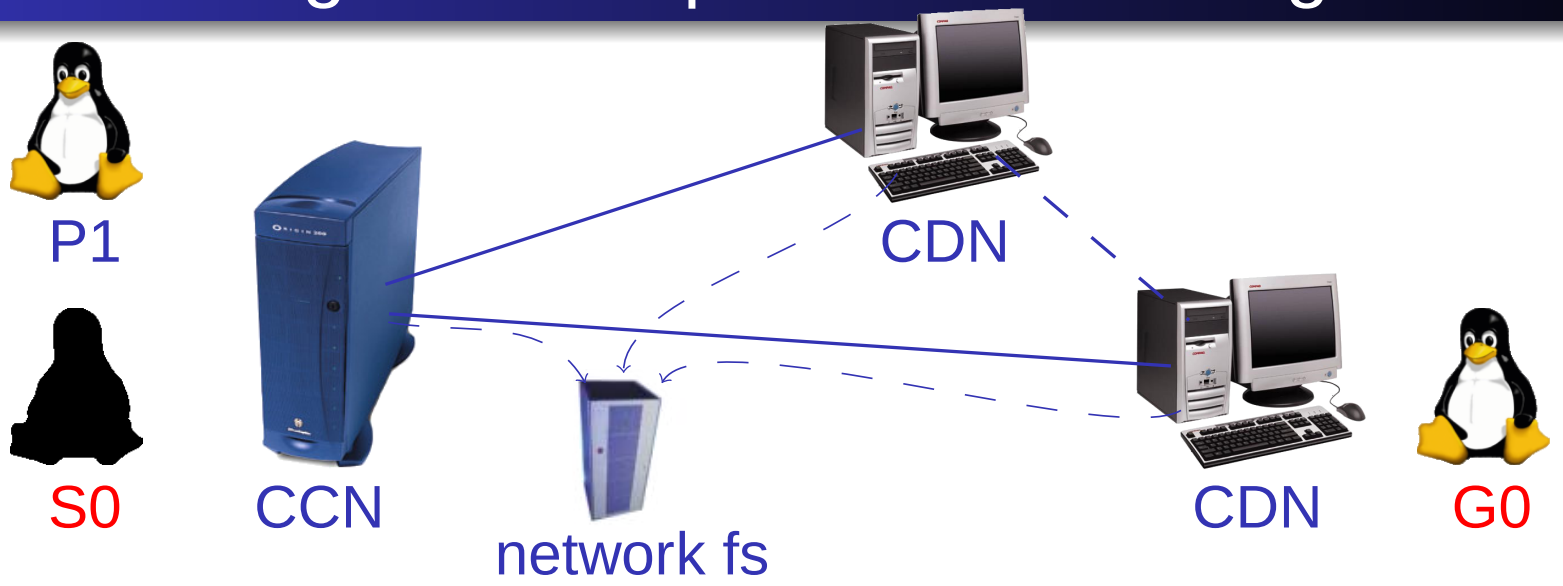
Process Migration Steps - inter-PEN Migration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_2)$.

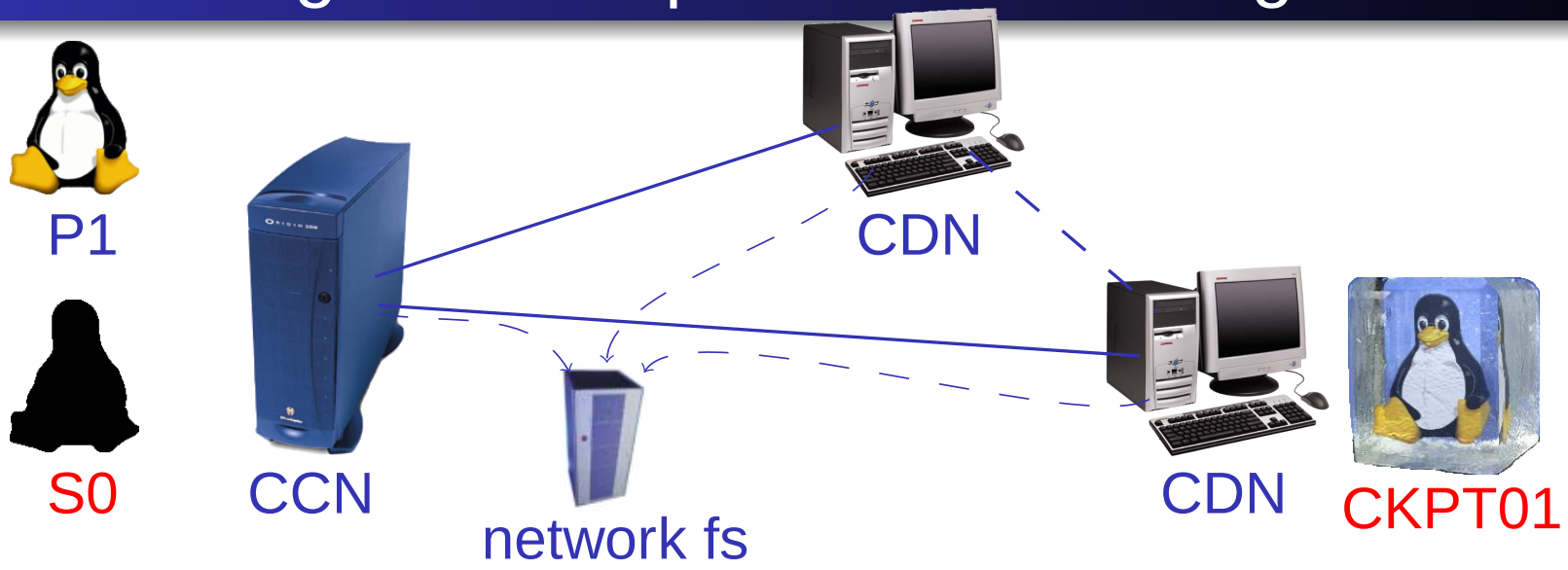
Process Migration Steps - inter-PEN Migration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_2)$.

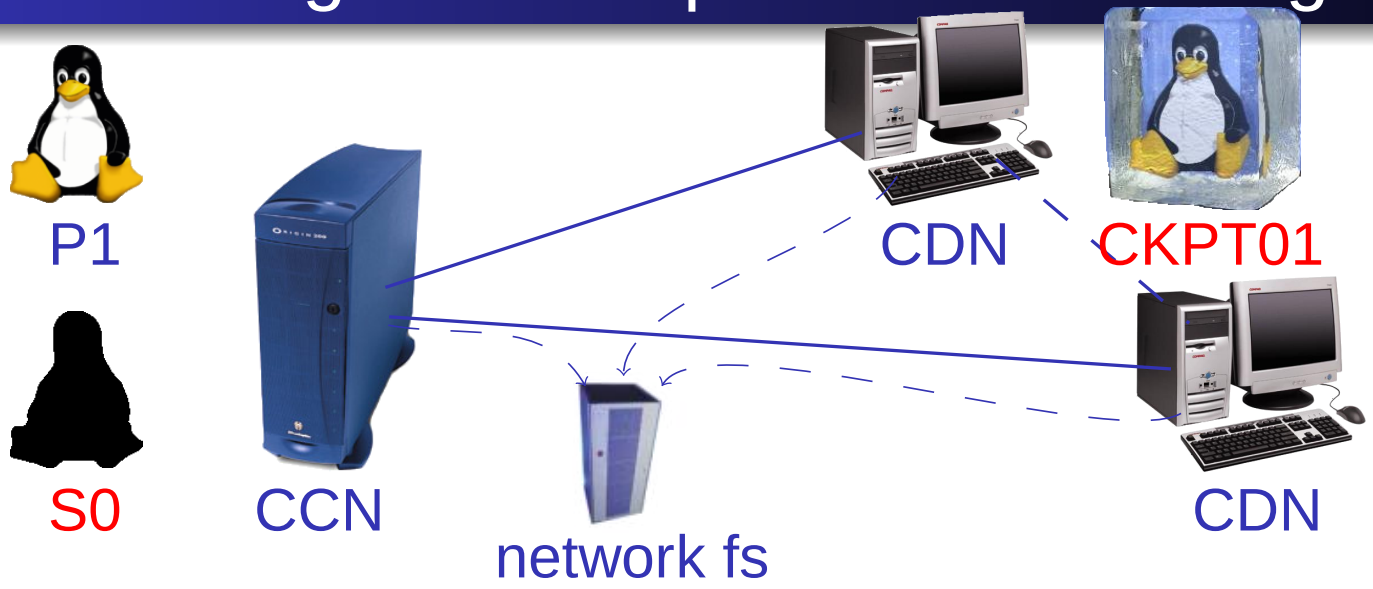
Process Migration Steps - inter-PEN Migration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_2)$.

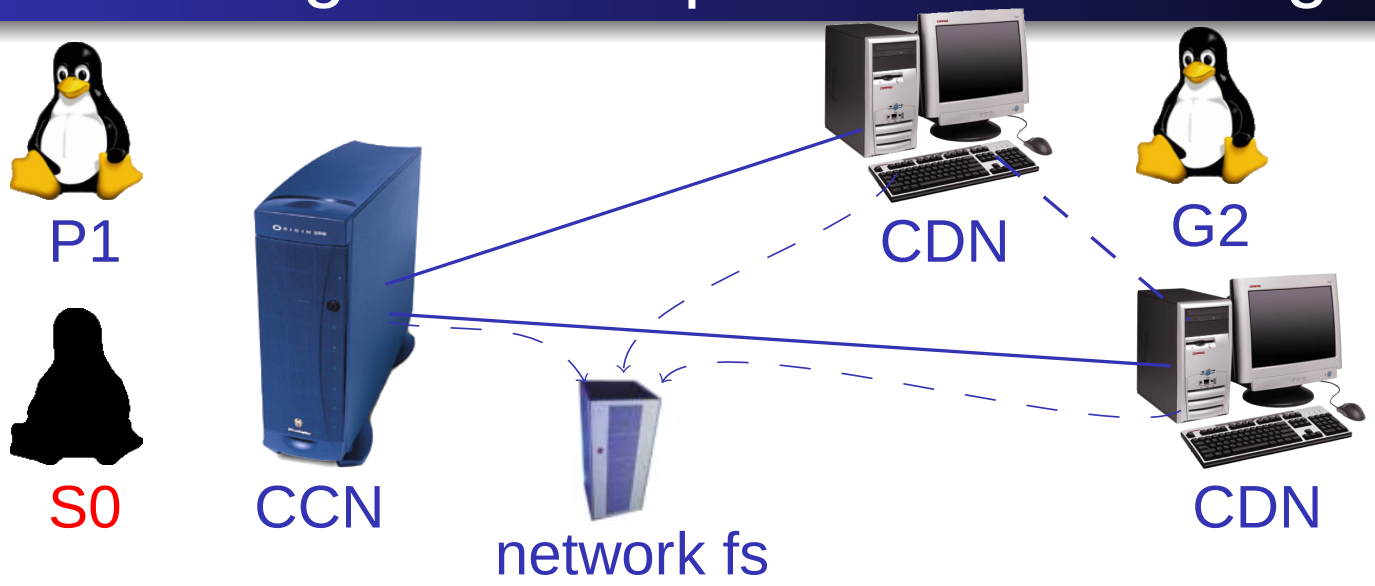
Process Migration Steps - inter-PEN Migration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_2)$.

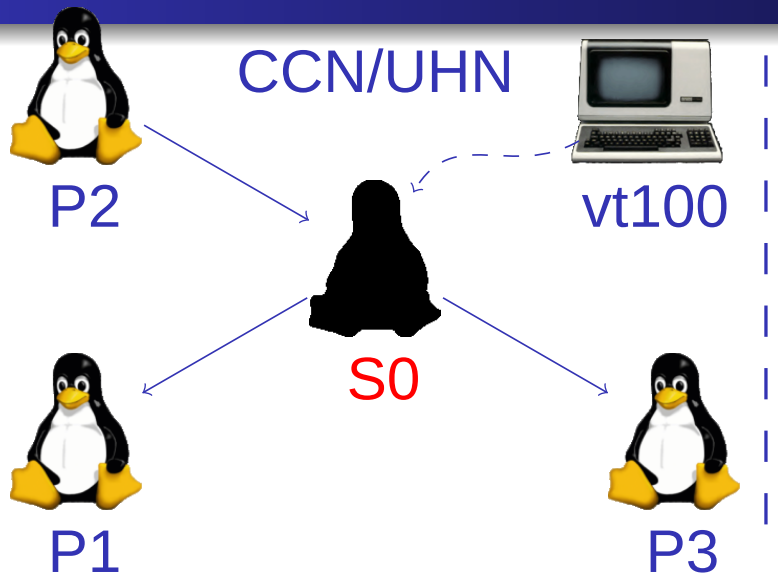
Process Migration Steps - inter-PEN Migration



Steps

1. Select a process (migration policy).
2. Freeze (checkpoint creation).
3. Move frozen process to the new execution node (migration).
4. Defrost (checkpoint restart) $P_0 \rightarrow (S_0, G_2)$.

Need for a Shadow and Guest Process



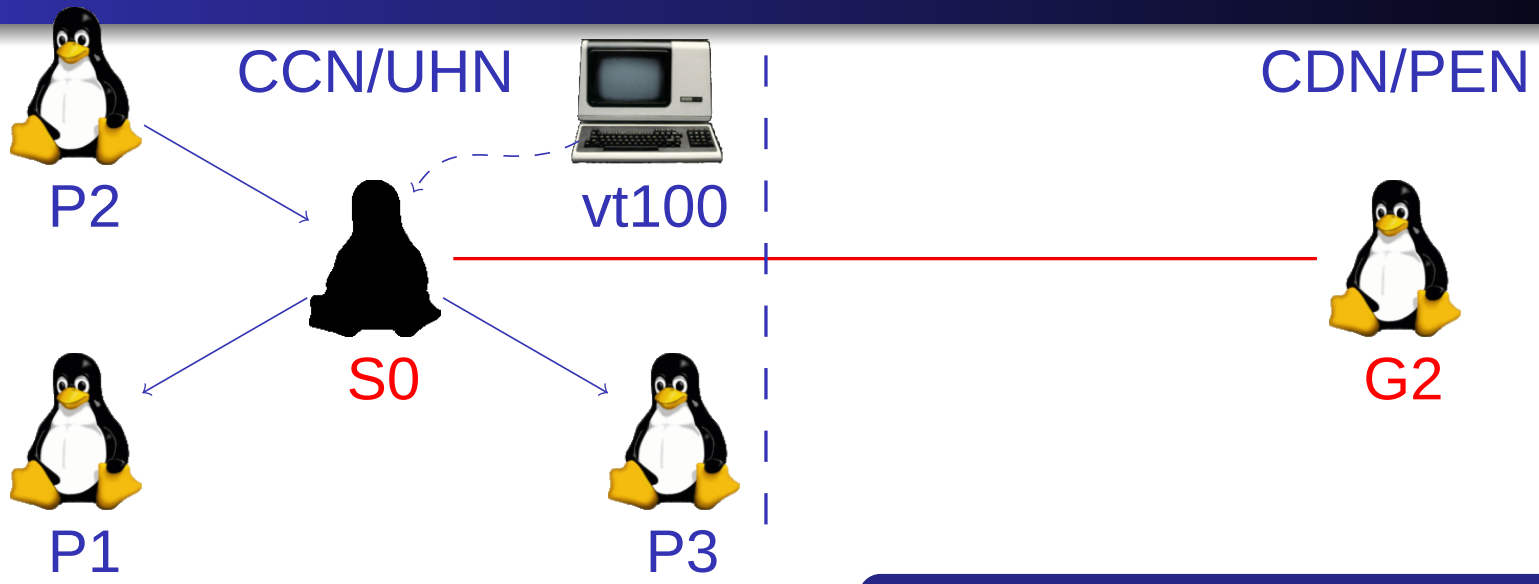
Why Shadow?

- Residual dependencies towards the home node.
- Migration slot for return.
- Communication with guest on PEN.

Why Guest?

- Migrated process abstraction on PEN.
- Use of provided resources.
- Communication with shadow
  - system call forwarding
  - signal forwarding

Need for a Shadow and Guest Process



Why Shadow?

- Residual dependencies towards the home node.
- Migration slot for return.
- Communication with guest on PEN.

Why Guest?

- Migrated process abstraction on PEN.
- Use of provided resources.
- Communication with shadow
  - system call forwarding
  - signal forwarding

TCMI - Requirements

Functional

- Mechanism for process checkpointing.
- Migration mechanism built on top of checkpointing.

System

- Target platform Linux kernel 2.6.x.
- Minimum footprint of kernel modifications.

Performance

- Minimize impact on local processes(system slowdown).
- Quickly resume execution of a migrating process.

TCMI - Requirements

Functional

- Mechanism for process checkpointing.
- Migration mechanism built on top of checkpointing.

System

- Target platform Linux kernel 2.6.x.
- Minimum footprint of kernel modifications.

Performance

- Minimize impact on local processes(system slowdown).
- Quickly resume execution of a migrating process.

TCMI - Requirements

Functional

- Mechanism for process checkpointing.
- Migration mechanism built on top of checkpointing.

System

- Target platform Linux kernel 2.6.x.
- Minimum footprint of kernel modifications.

Performance

- Minimize impact on local processes(system slowdown).
- Quickly resume execution of a migrating process.

TCMI - Features

- Fully OO design and implementation, SMP safe.
- Business logic platform(OS) independent.

Component Oriented Framework

- Control filesystem.
- KKC sockets.
- Communication component built on KKC.
  - Asynchronous message exchange
  - Synchronous message exchange(transactions)
- Node management components.
- Checkpointing component.
- Migration component.
- Shadow/Guest process abstraction.

TCMI - Features

- Fully OO design and implementation, SMP safe.
- Business logic platform(OS) independent.

Component Oriented Framework

- Control filesystem.
- KKC sockets.
- Communication component built on KKC.
  - Asynchronous message exchange
  - Synchronous message exchange(transactions)
- Node management components.
- Checkpointing component.
- Migration component.
- Shadow/Guest process abstraction.

TCMI - Features

- Fully OO design and implementation, SMP safe.
- Business logic platform(OS) independent.

Component Oriented Framework

- Control filesystem.
- KKC sockets.
- Communication component built on KKC.
  - Asynchronous message exchange
  - Synchronous message exchange(transactions)
- Node management components.
- Checkpointing component.
- Migration component.
- Shadow/Guest process abstraction.

TCMI - Features

- Fully OO design and implementation, SMP safe.
- Business logic platform(OS) independent.

Component Oriented Framework

- Control filesystem.
- KKC sockets.
- Communication component built on KKC.
  - Asynchronous message exchange
  - Synchronous message exchange(transactions)
- Node management components.
- Checkpointing component.
- Migration component.
- Shadow/Guest process abstraction.

TCMI - Features

- Fully OO design and implementation, SMP safe.
- Business logic platform(OS) independent.

Component Oriented Framework

- Control filesystem.
- KKC sockets.
- Communication component built on KKC.
  - Asynchronous message exchange
  - Synchronous message exchange(transactions)
- Node management components.
- Checkpointing component.
- Migration component.
- Shadow/Guest process abstraction.

TCMI - Features

- Fully OO design and implementation, SMP safe.
- Business logic platform(OS) independent.

Component Oriented Framework

- Control filesystem.
- KKC sockets.
- Communication component built on KKC.
  - Asynchronous message exchange
  - Synchronous message exchange(transactions)
- Node management components.
- Checkpointing component.
- Migration component.
- Shadow/Guest process abstraction.

TCMI - Features

- Fully OO design and implementation, SMP safe.
- Business logic platform(OS) independent.

Component Oriented Framework

- Control filesystem.
- KKC sockets.
- Communication component built on KKC.
  - Asynchronous message exchange
  - Synchronous message exchange(transactions)
- Node management components.
- Checkpointing component.
- Migration component.
- Shadow/Guest process abstraction.

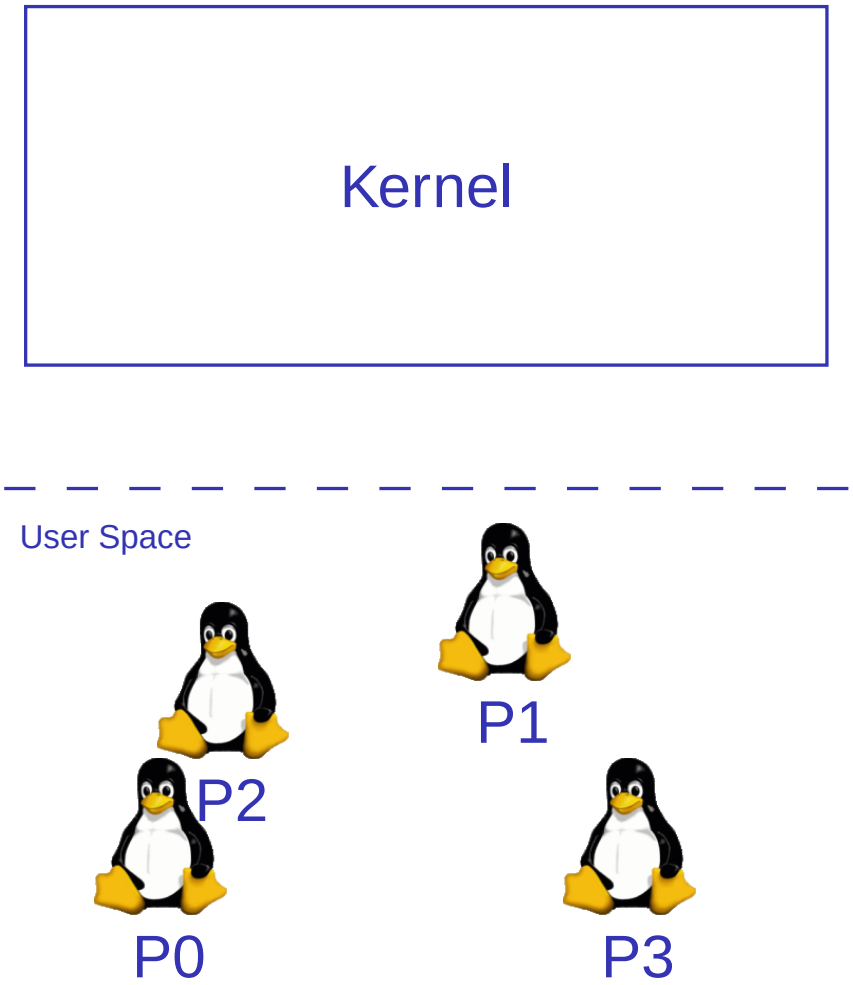
TCMI - Features

- Fully OO design and implementation, SMP safe.
- Business logic platform(OS) independent.

Component Oriented Framework

- Control filesystem.
- KKC sockets.
- Communication component built on KKC.
  - Asynchronous message exchange
  - Synchronous message exchange(transactions)
- Node management components.
- Checkpointing component.
- Migration component.
- Shadow/Guest process abstraction.

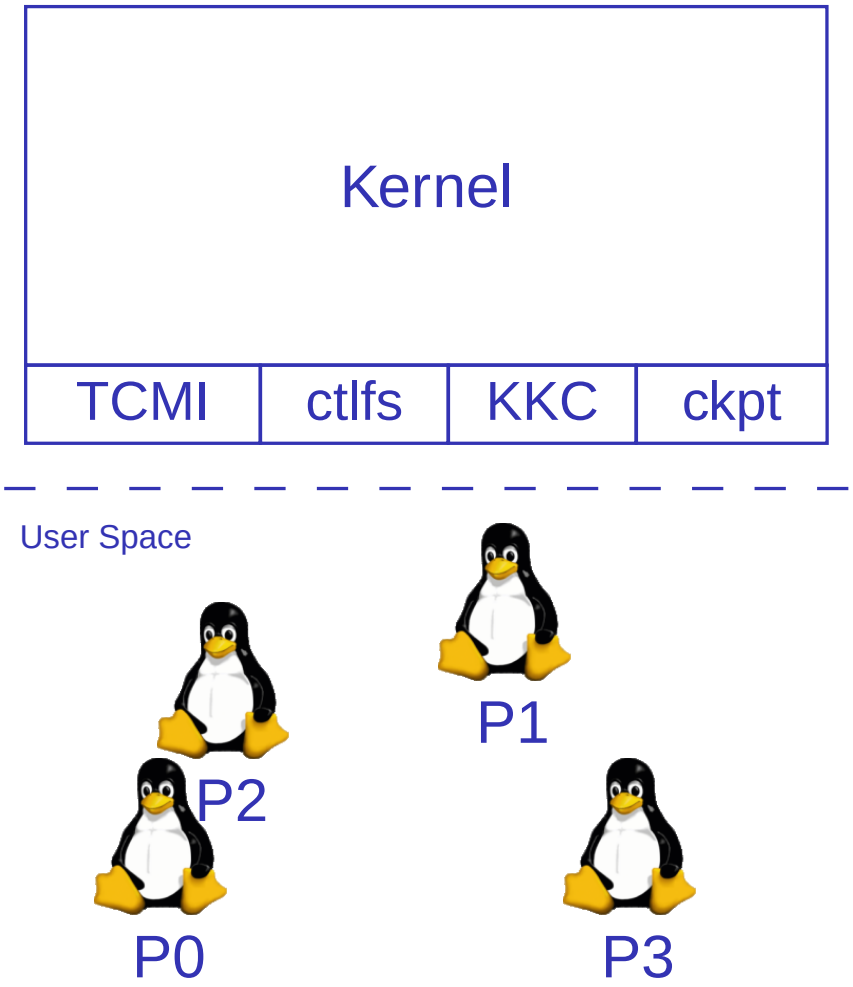
TCMI - Integration within OS kernel



Integration Structure

- Linux kernel
  - monolithic
  - loadable modules
- TCMI modules
  - linked on fly
- TCMI hooks
  - process descriptor
  - new kernel signal(`SIGUNUSED`)
  - system calls (`exit()`)

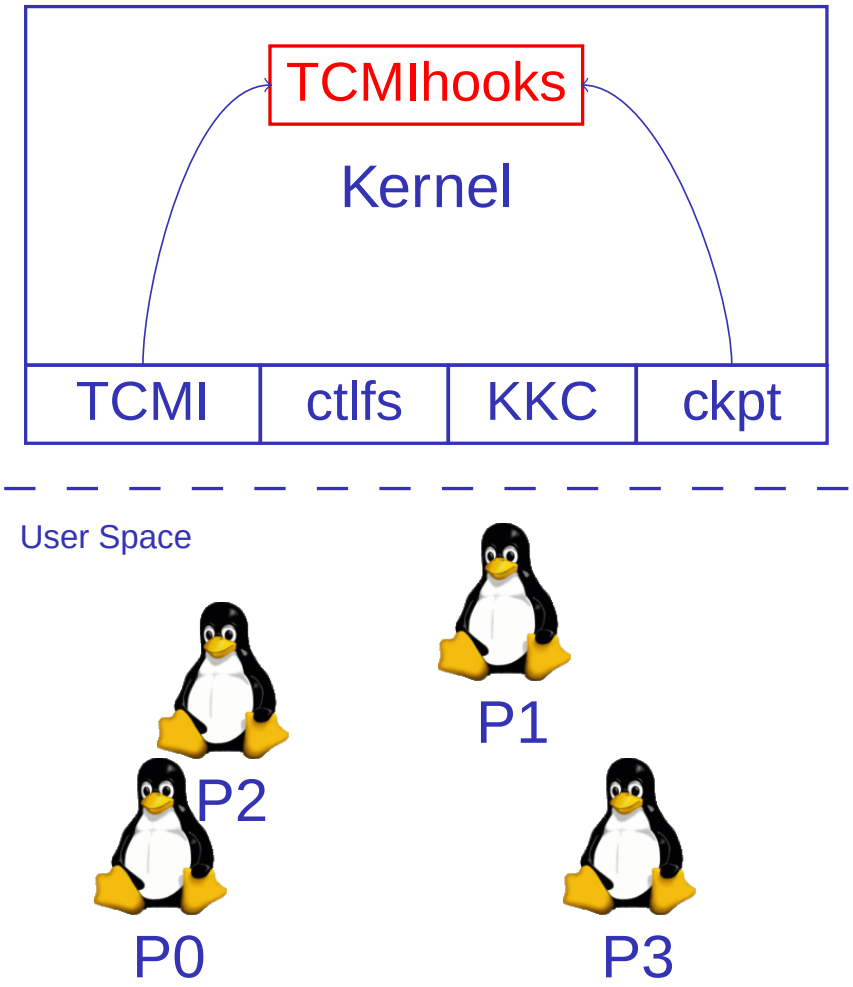
TCMI - Integration within OS kernel



Integration Structure

- Linux kernel
  - monolithic
  - loadable modules
- TCMI modules
  - linked on fly
- TCMI hooks
  - process descriptor
  - new kernel signal(`SIGUNUSED`)
  - system calls (`exit()`)

TCMI - Integration within OS kernel



Integration Structure

- Linux kernel
  - monolithic
  - loadable modules
- TCMI modules
  - linked on fly
- TCMI hooks
  - process descriptor
  - new kernel signal(`SIGUNUSED`)
  - system calls (`exit()`)

Summary

TCMI

- Component oriented OO puzzle.
- Preemptive process migration and checkpointing.
- 7 main components in 45 classes.
- Node management capabilities.
- Provides mechanism not policy.
- 8200 SLOC base code + 2500 SLOC in test cases.

Future Work

- Migration via virtual checkpoint image.
- Inter-PEN process migration.
- Migration of LWP's(threads).
- Finish system call/signal forwarding.

Questions

?

Typeset with LaTeX 2ε and Beamer