Revisiting Virtual Memory for High Performance Computing on Manycore Architectures: A Hybrid Segmentation Kernel Approach

Yuki Soma, Balazs Gerofi, Yutaka Ishikawa

Agenda

- Background on virtual memory
- Design & Implementation of the hybrid segmentation kernel approach
- Evaluation
- Related Work
- Conclusion & Future Work

Two Types of Virtual Memory

Cause of memory fragmentation

Paging

Segmentation

Completely ignored today!

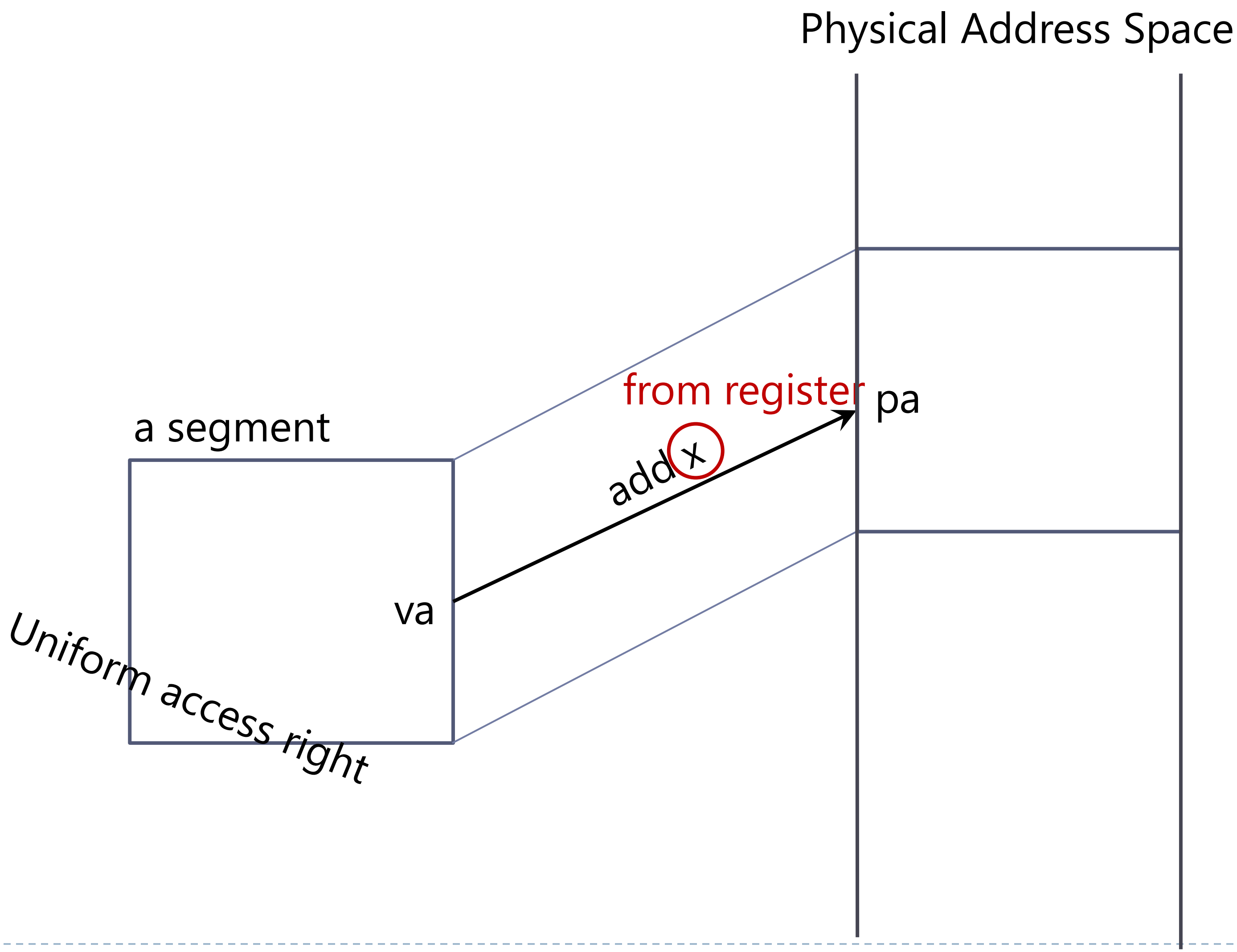
Too coarse access control

Is Paging Really Better?

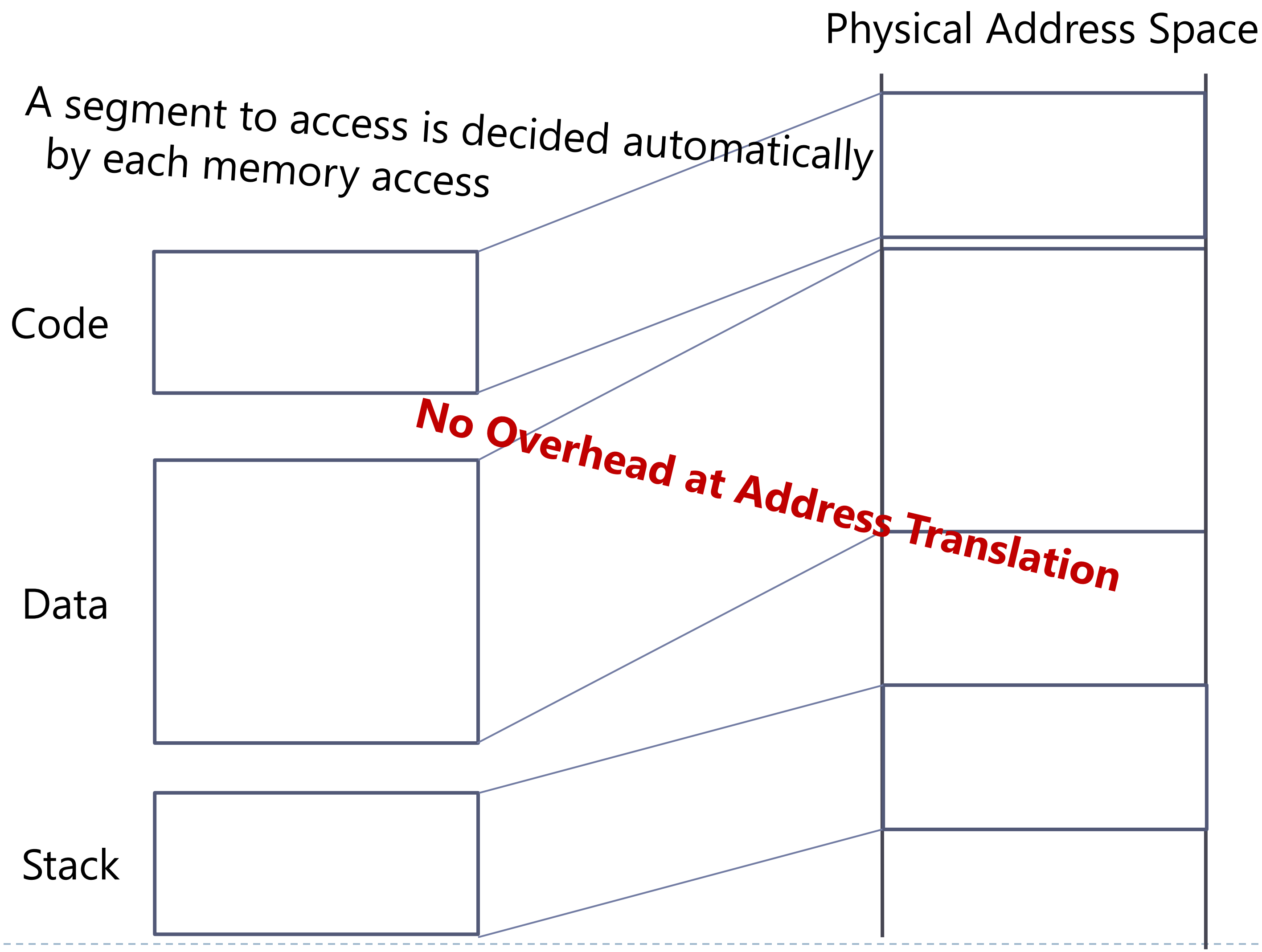
Cost of paging is high

- Paging degrades performance
  - Accounts for **50**% of execution time [McCurdy et al., 08]
- Paging costs energy
  - Accounts for **3-14**% of CPU core power [Sodani, 11]
- **It will get higher in the future!**
  - Emergence of data-centric workloads [Ranganathan, 11]
  - Manycore trend -> TLB shootdown
    - Invalidation of TLBs by software to keep TLBs consistent
    - **Over 10**% (tens of cores) at some app.s [Romanescu et al, 10]

Usage of Segmentation in x86



Usage of Segmentation in x86

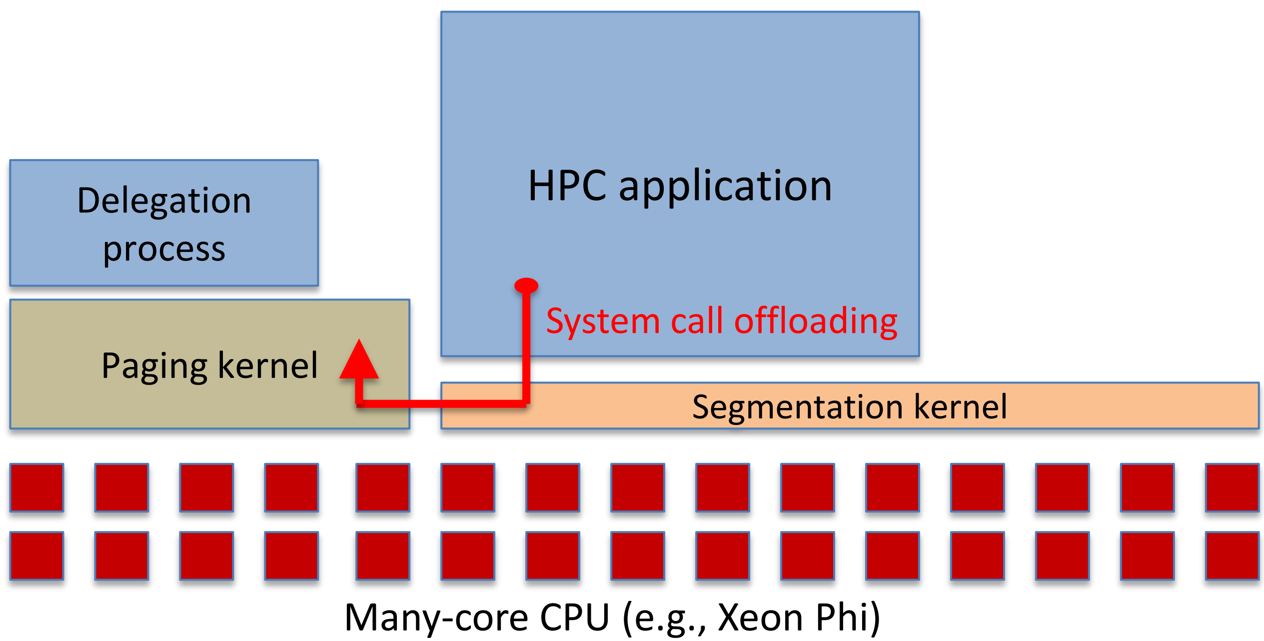


Agenda

- ✓ Background on virtual memory
- Design & Implementation of the hybrid segmentation kernel approach
- Evaluation
- Related Work
- Conclusion & Future Work

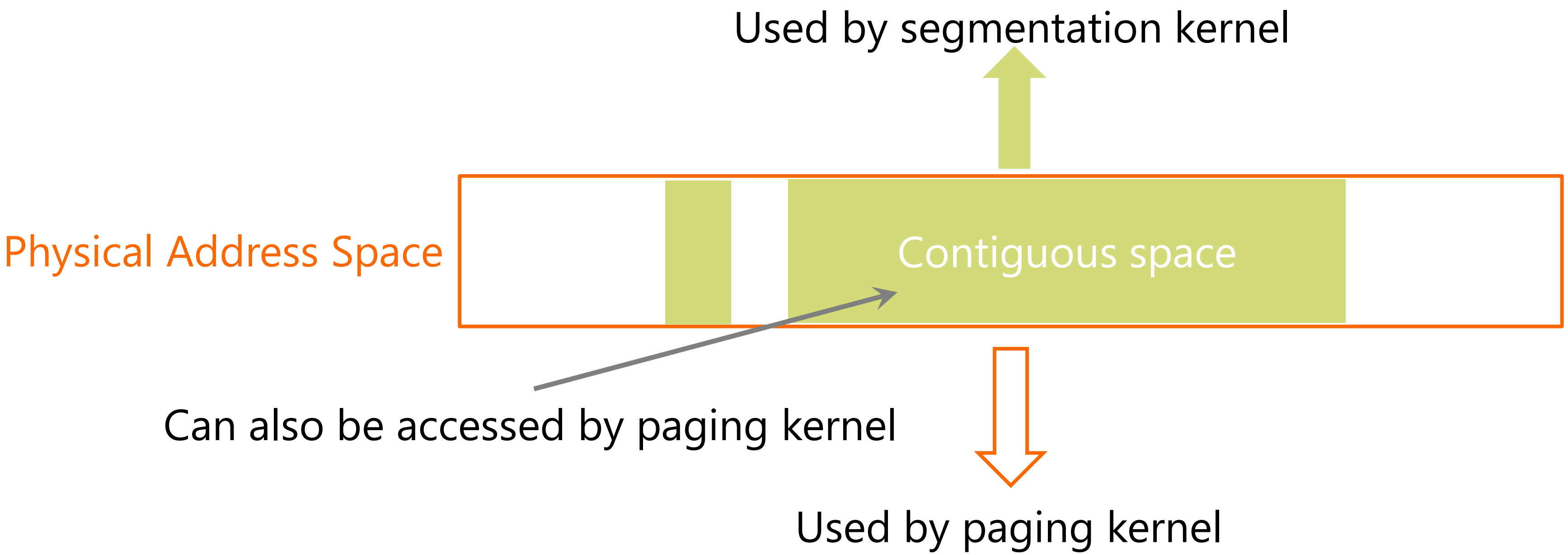
Hybrid Approach

- Our Approach: **Mix the two mechanisms**
  - Paging/Segmentation can be set at each core independently in x86
  - Segmentation kernel is too small to handle all system calls



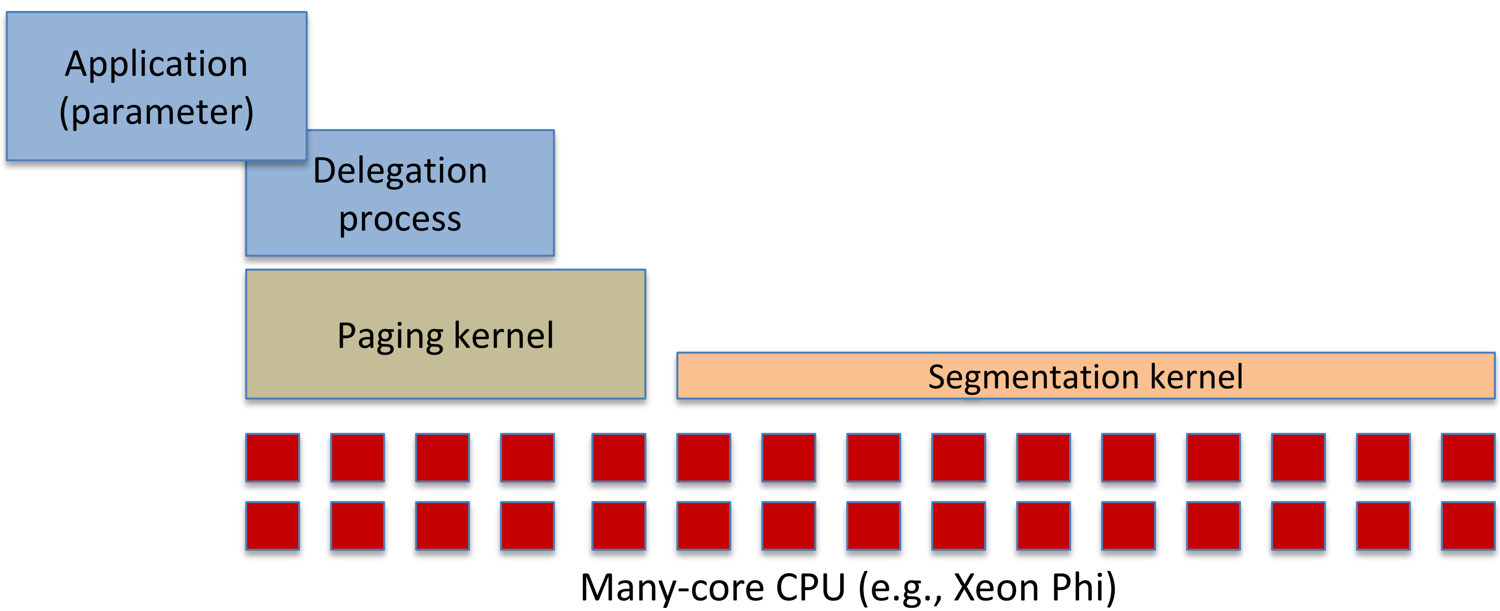
Memory mapping

- Contiguous spaces are reserved as segments
  - For the segmentation kernel and applications
  - Also for the communication between the two kernels
- Other parts are used only by paging kernel

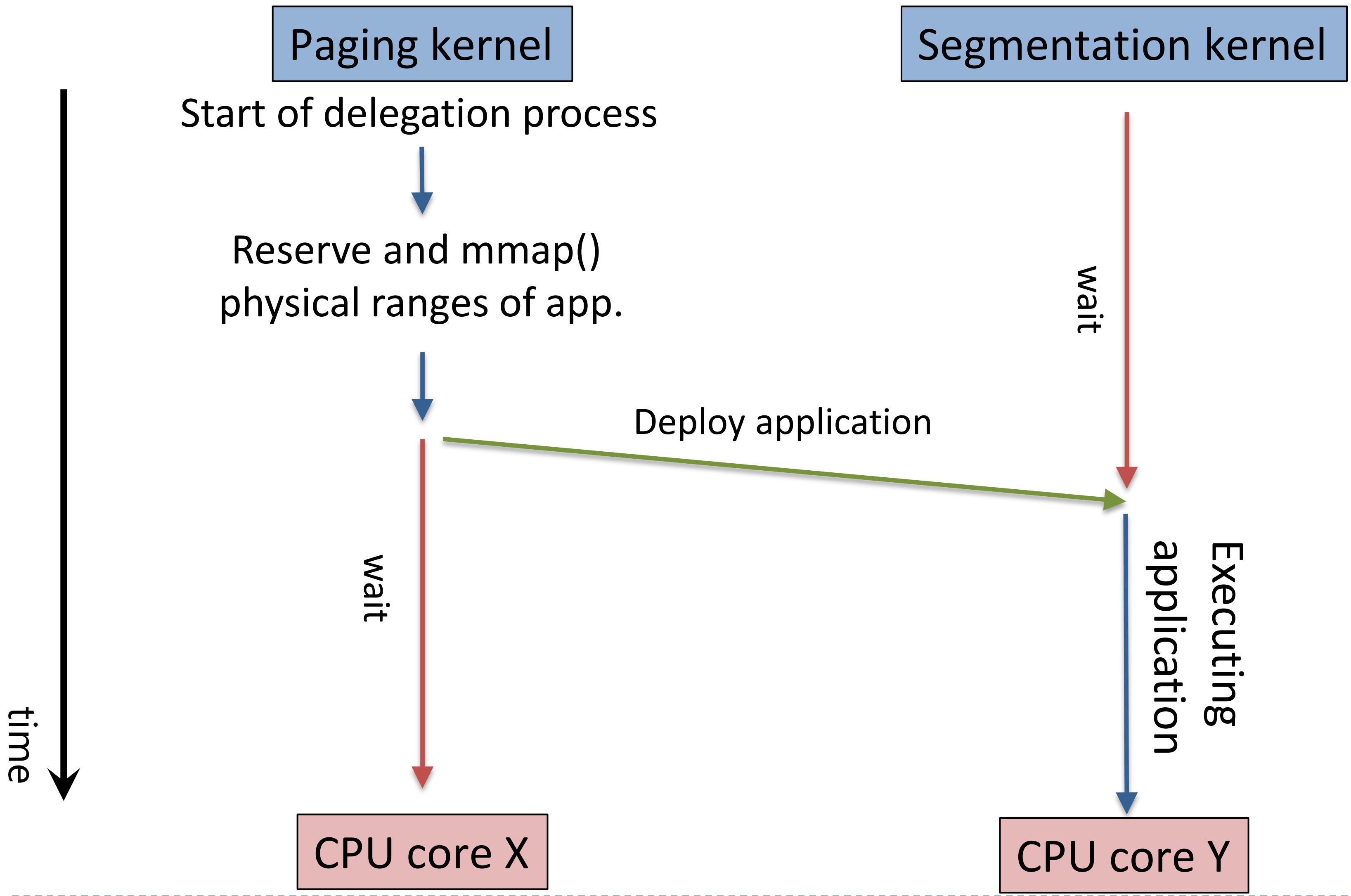


Delegation Program

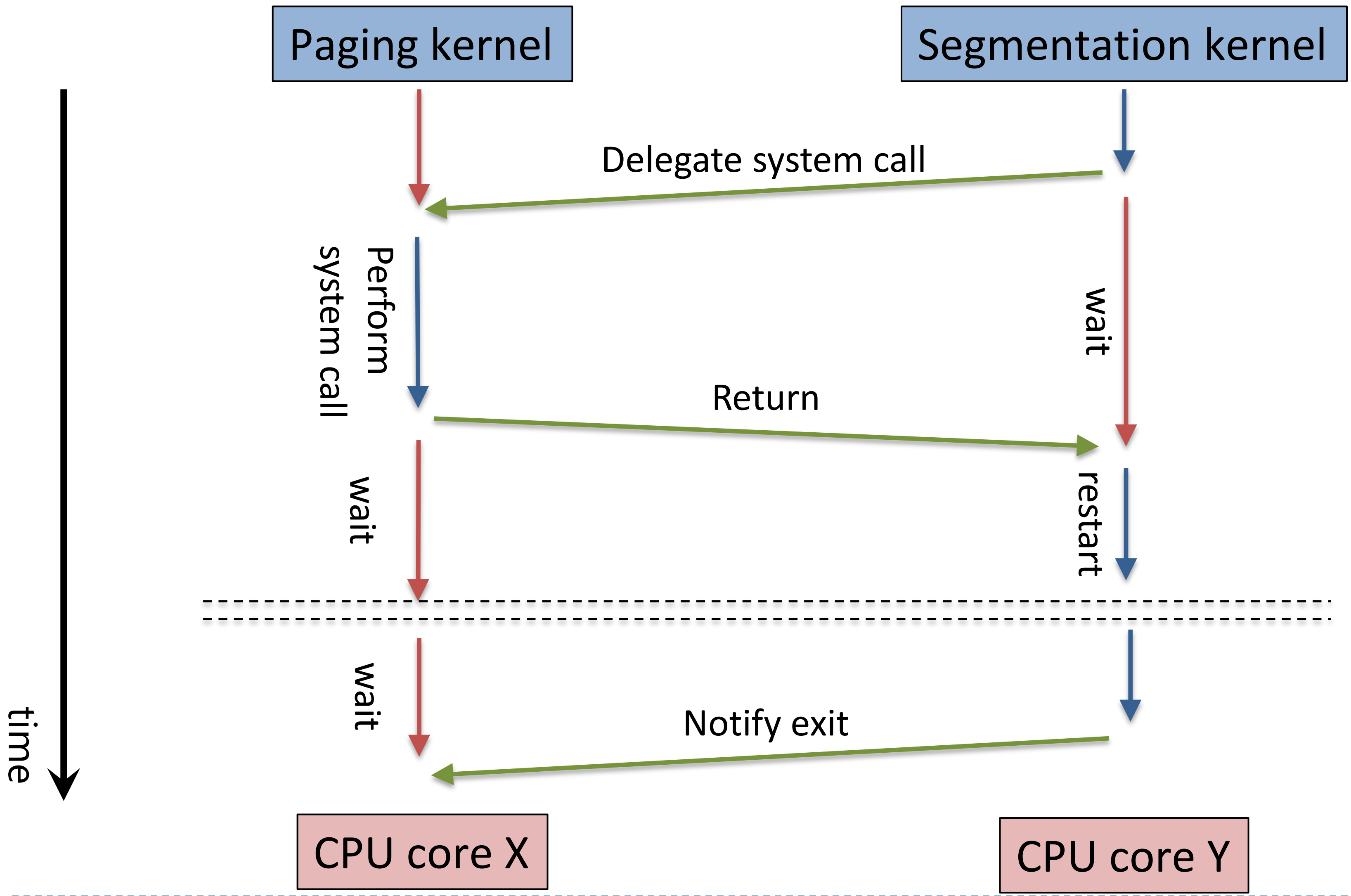
- Executed at paging cores
- Deploys the application to segmentation kernel
- Waits for system call offloading



Flow Chart of Execution



Flow Chart of Execution



Advantage

- **Completely eliminates paging cost**
  - Page walk (address translation) cost
  - Overhead to TLB shootdown
  - TLB power consumption…
- OS kernel can still use paging features
- Implementation is not so difficult
  - We can use system call handlers in paging kernel's code

Limitation

- ≒ **Limitation of segmentation (for applications)**
  - We can't change the size of (data) segment
  - Internal memory fragmentation
  - No access control (only read/write for data segment)

- Characteristics of applications are important
  - No too complicated memory (de)allocation patterns
  - No need of access control

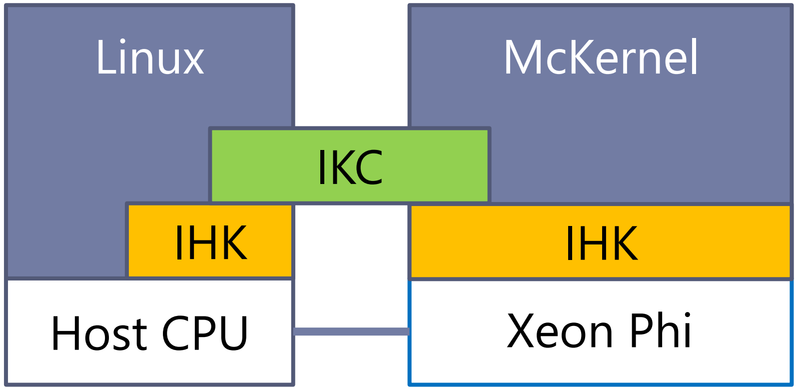
Restrictions of Implementation

- We had to use **32 bit mode**!
  - Segmentation is not fully supported in x86's 64 bit mode
  - ⇒ 32-bit segmentation kernel & 64-bit paging kernel
  - Memory usage ≦ 4GB
- Currently few system calls are supported
  - e.g. neither fork() nor clone() are supported

McKernel

- Used as the paging kernel (running on one core)
- A light-weight kernel for manycore architecture
  - Developed by U-Tokyo, RIKEN AICS, Hitachi, NEC, and Fujitsu
- Running with the help of host Linux
  - Program execution on McKernel through host Linux
  - System call delegation to host Linux

IKC: Inter Kernel Communication
IHK: Interface for Heterogeneous Kernels



Two system calls added to McKernel

- These are called by the delegation process

- *init_core()*
  1. Boot segmentation cores
  2. Reserve & mmap() contiguous address space

- *load_core()* (the main part of delegation process)
  1. Load the binary of the application into the reserved area
  2. Start the application and wait
  3. Perform delegated system calls
  4. Exit when the application exits

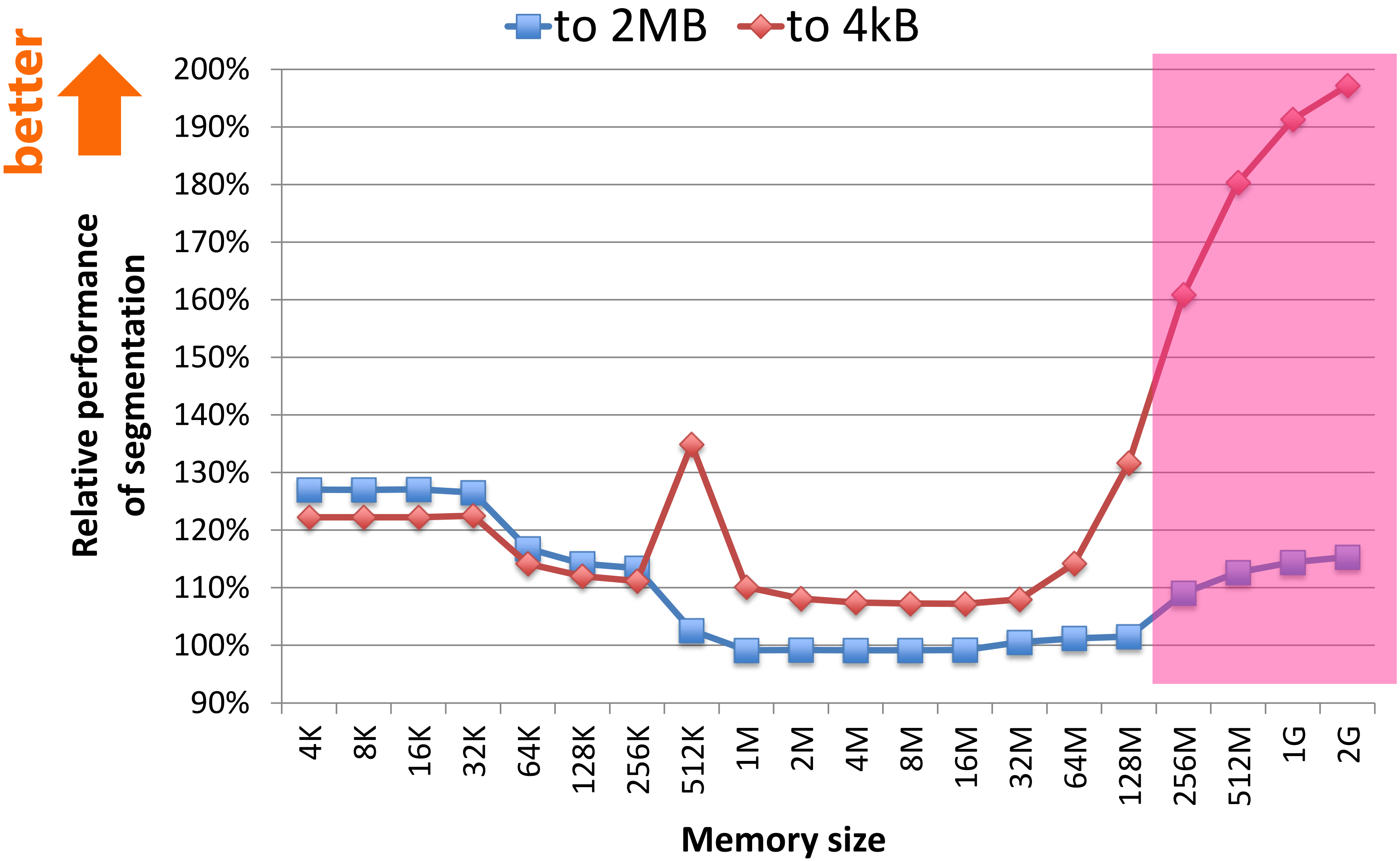
Agenda

- ✓ Background on virtual memory
- ✓ Design & Implementation of the hybrid segmentation kernel approach
- Evaluation
- Related Work
- Conclusion & Future Work

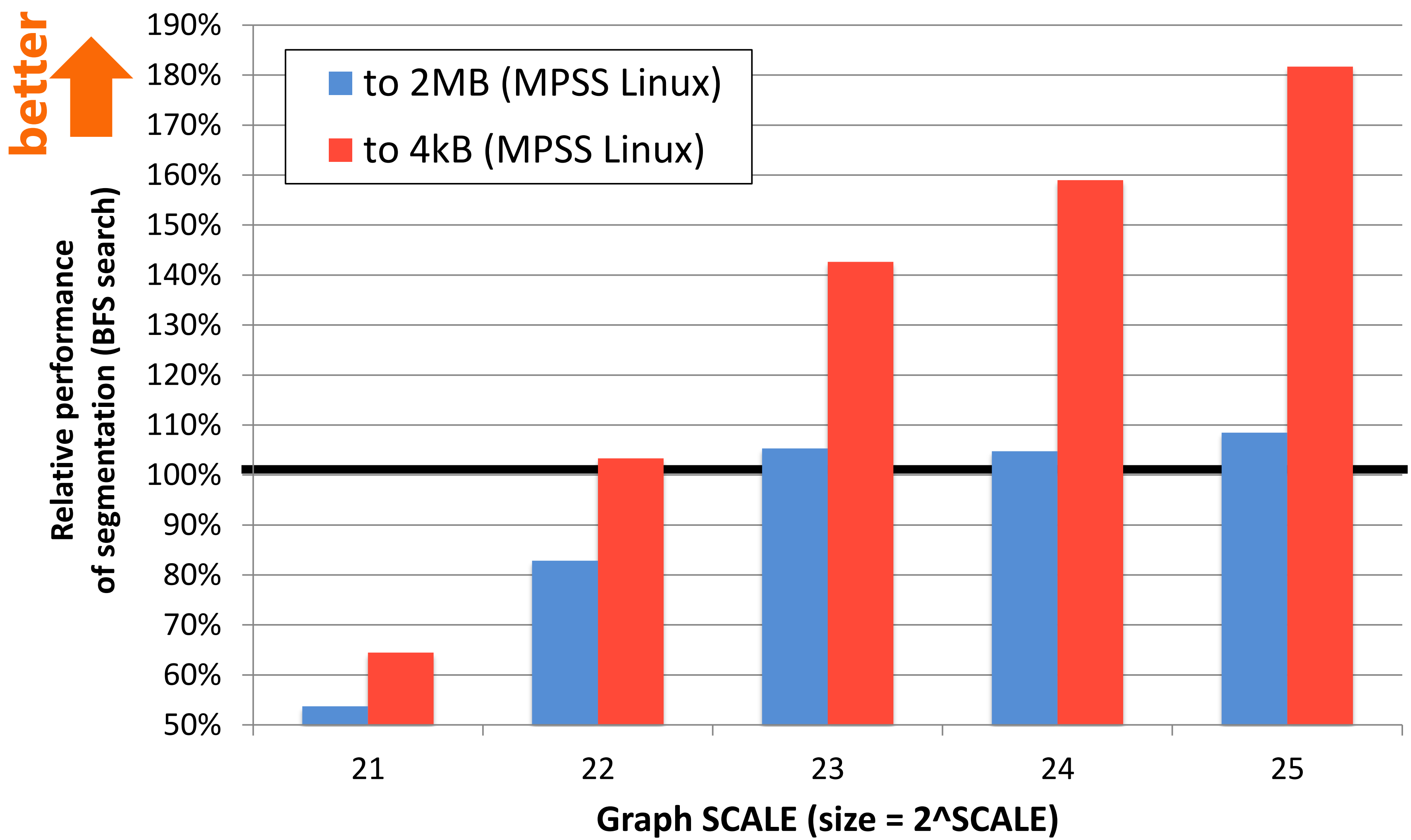
Evaluation

- RandomAccess in the HPC Challenge benchmark
  - Our Approach **vs. McKernel** with 4K/2MB pages
  - Codes are almost equivalent at binary level
- Graph500
  - Our Approach **vs. MPSS Linux** with 4K/2MB pages
  - **Codes are very different** at binary level
    - Of course the same source code
    - *i.e.* maximum load/store size, special instructions
- Both of them are executed on single thread

RandomAccess



Graph500



better
Relative performance
of segmentation (BFS search)
190%
180%
170%
160%
150%
140%
130%
120%
110%
100%
90%
80%
70%
60%
50%
to 2MB (MPSS Linux)
to 4kB (MPSS Linux)
21
22
23
24
25
Graph SCALE (size = 2^SCALE)

Agenda

- ✓ Background on virtual memory
- ✓ Design & Implementation of the hybrid segmentation kernel approach
- ✓ Evaluation
- Related Work
- Conclusion & Future Work

Related Work

- Direct segment [Basu et al, 13]
  - New hardware to combine paging and segmentation
  - They only give performance <u>estimation</u>

- FusedOS [Park et al, 12]
  - Applications run on a light-weight kernel
    - with system call offloading to Linux
  - Not address the TLB issues

- Other research on paging
  - Not eliminate paging cost completely

Agenda

- ✓ Background on virtual memory
- ✓ Design & Implementation of the hybrid segmentation kernel approach
- ✓ Evaluation
- ✓ Related Work
- Conclusion & Future Work

Conclusion

- Hybrid kernel approach on manycore architecture
  - Most cores are in segmentation, some cores are in paging
  - Applications runs over segmentation
  - System call offloading to paging kernel

- It gets **81**% (4KB page) and **9**% (2MB page) improvement compared to a OS based on paging
  - Graph500

- We encourage hardware designers to consider full support of segmentation in x86 64-bit mode!

Future Work

- Support for multi-threading

- Evaluation in terms of OS noise
  - Reduction of OS noise -> performance predictability?