Accelerating Encrypted Storage with OpenCL Devices

Te-Sheng Lin and Chia-Heng Tu

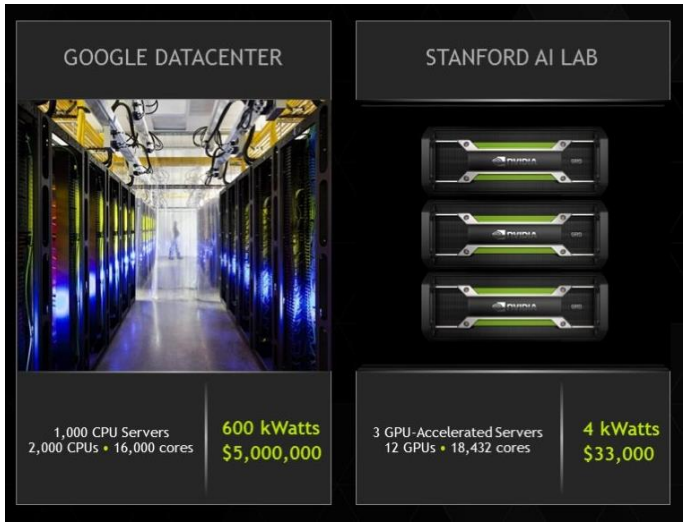
National Cheng Kung University, Taiwan

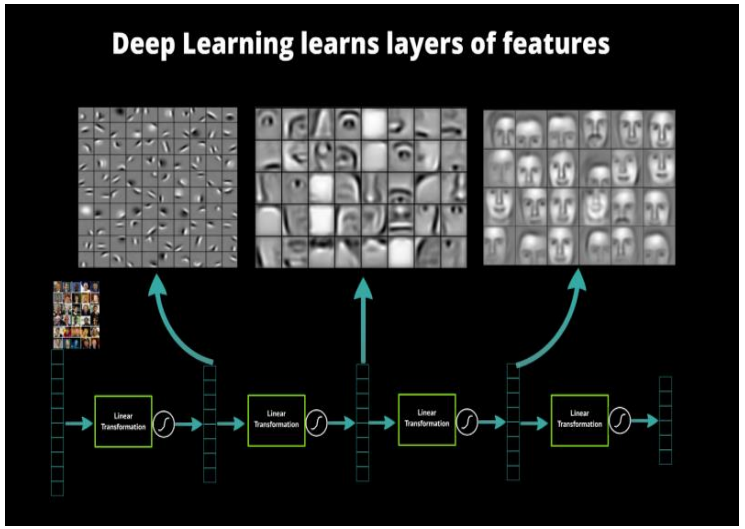
USENIX FAST'18 WIP Feb.2018

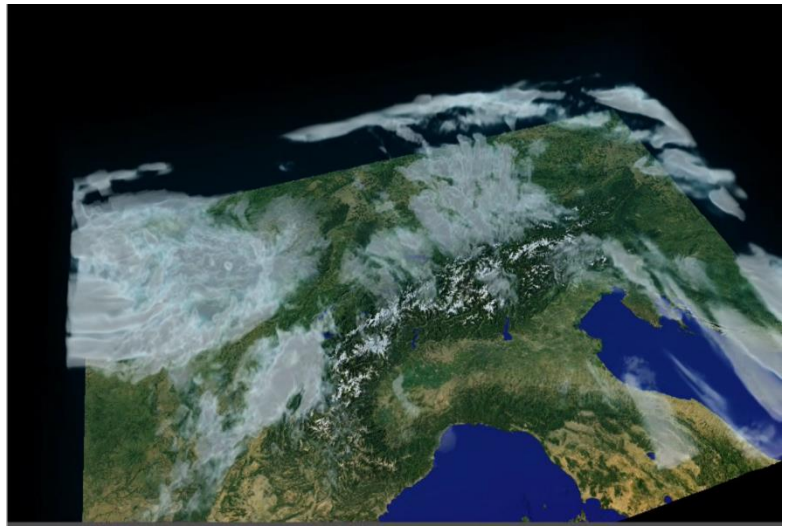


Background

- **Heterogeneous computing is widely adopted in various user-space applications**
  - Different types of accelerators are often used to boost the performance.
  - Compute-Hungry applications: Machine Learning, Weather Forecasting.
- **Similarly, Linux kernel-space software demands more computing power**
  - Data encryption, hashing, and network packet analysis.

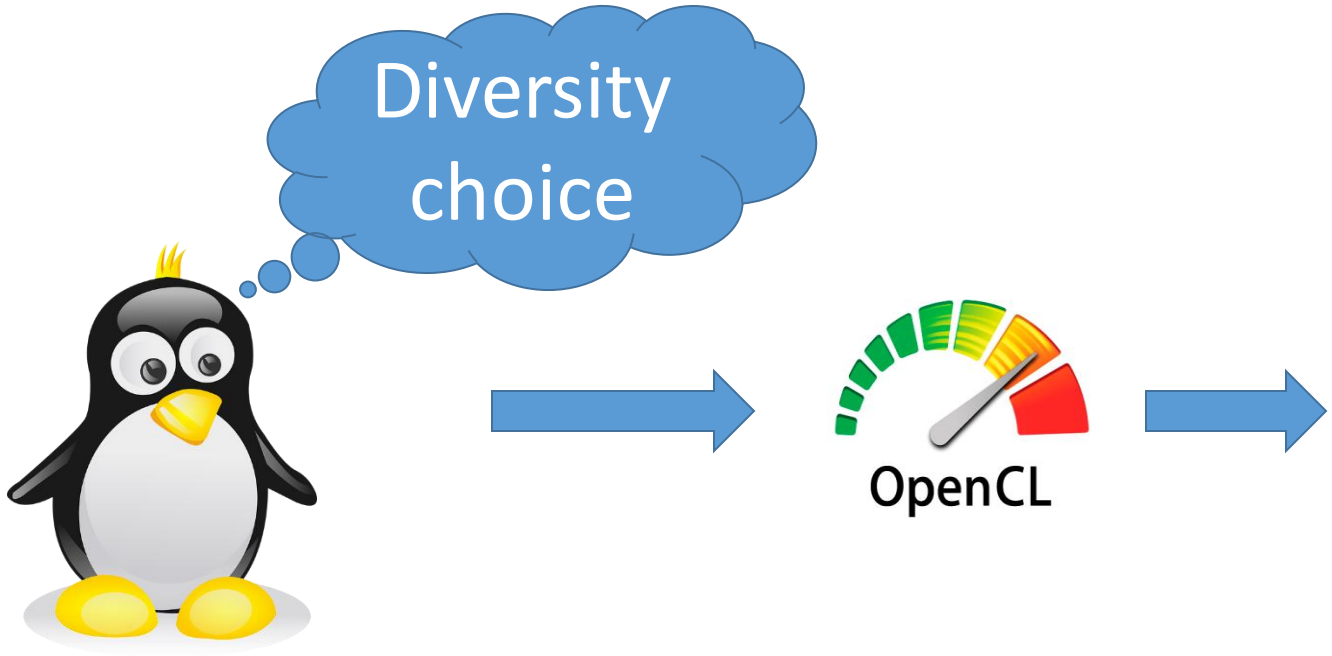






Our Proposal: KOCL Framework

- Linux **K**ernel acceleration with **O**pen**CL**.
- **OpenCL** is chosen for:
  - its support various types of accelerators: **multi-core CPUs, GPUs, FPGA & DSP.**
  - its widely adoptions by hardware vendors.

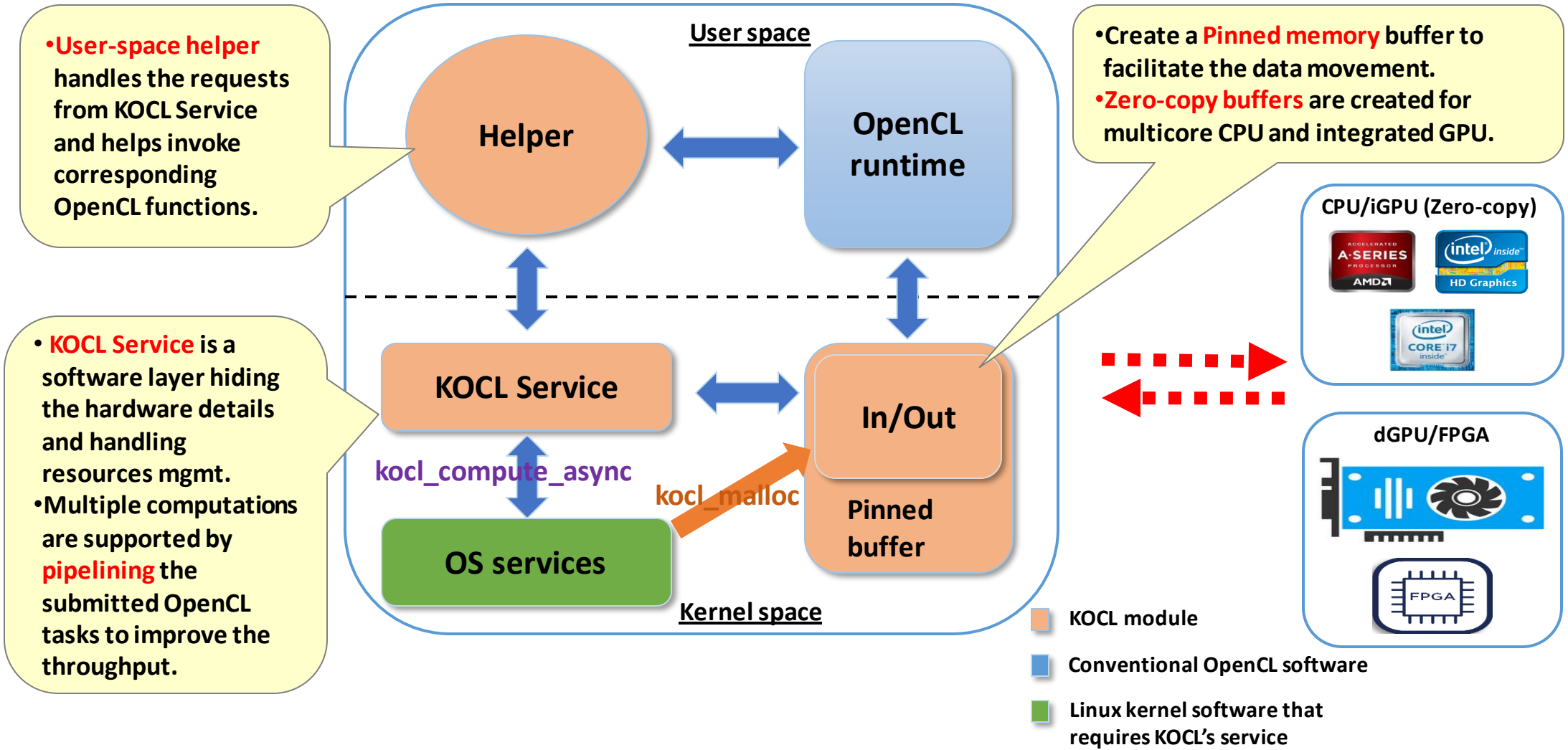




Challenges for In-Kernel Acceleration

1. **OpenCL** is a **user-space runtime** library
   - By default, **kernel-space** software **cannot invoke** the **user-space function** calls directly.
   - Porting the entire OpenCL software stack into kernel-space is cumbersome and non-portable.
2. **Programming** overhead
   - OpenCL offers **low-level APIs** for parallel computing.
   - It requires the programmers to take into account of the **architecture- and platform-dependent issues**, such as memory architectures, for high performance.
3. **Data allocation and movement** problem
   - In conventional OpenCL program development, **moving** user-space data in between **Host** and **Device** memory should be treated delicately for high performance.
   - Moving **kernel-space data forth and back** on top of conventional OpenCL software makes things (*data movements*) more complex.
4. **Multiple services** support
   - Things get harder when **multiple computations** in Linux kernel space needs to be served at the same time.
   - While OpenCL natively supports multiple tasks, it is hard for individual kernel module developer to do resource management properly without sacrificing another module's performance.

Key Components in KOCL

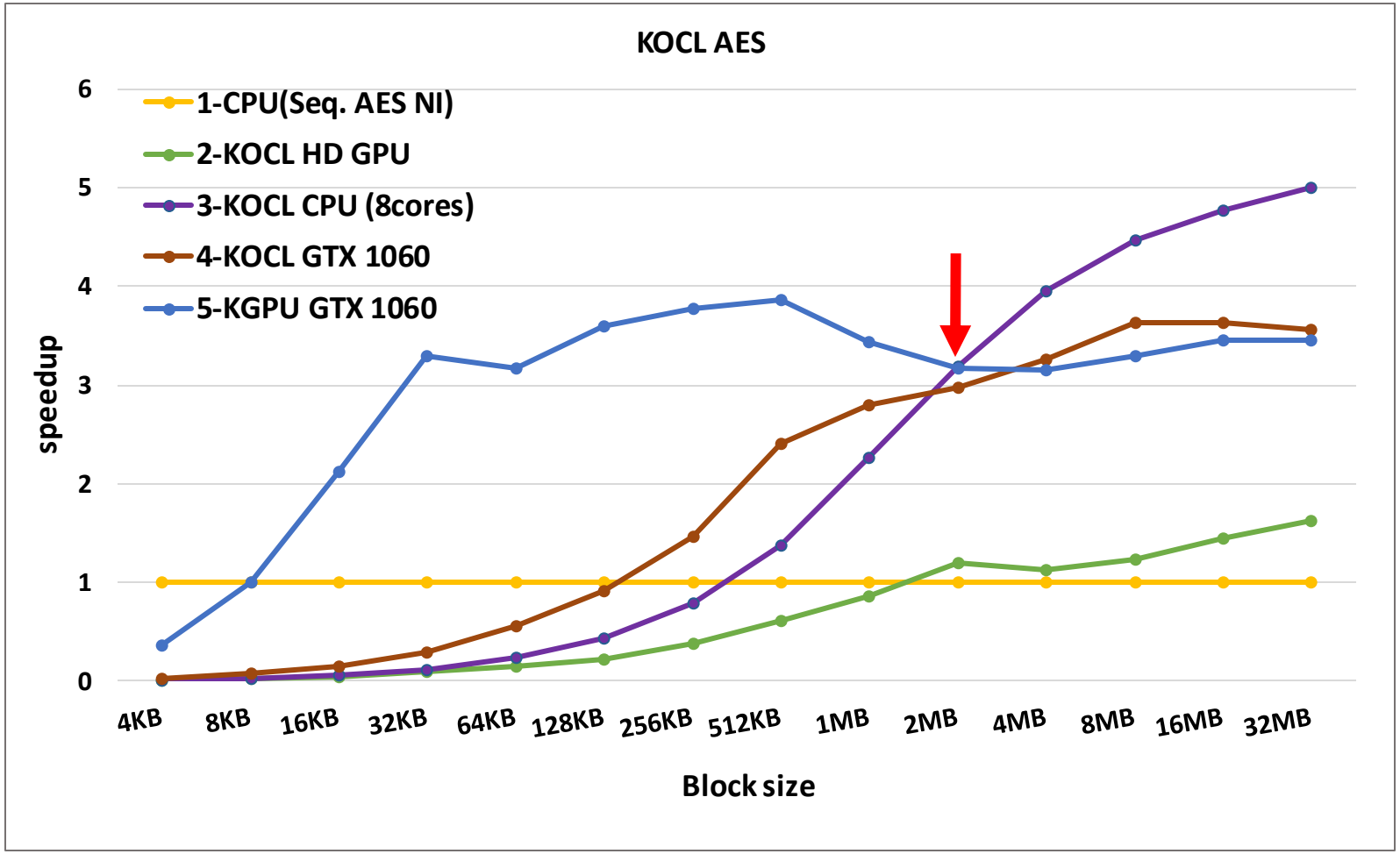


Experimental Setup

- **Software:**
  - **AES-NI** [Keon Jang, In *NSDI 2011*]
  - CUDA 9.0, OpenCL 2.0
  - Ubuntu 16.04LTS (Linux kernel v4.7.0)
- **Hardware (three OpenCL devices):**
  - **Intel i7-6700** (8 core @3.4GHz) w/ 16GB RAM
  - **Intel HD 530** Graphics (24 cores @1.15GHz)
  - **NVIDIA GTX 1060** GPU (1280 cores @1.7GHz) w/ 6GB onboard memory
- Compare **relative performance of the OpenCL devices** to process various sizes of input data, from **4KB** to **32MB**
  - **KGPU's** [Sun. *In SYSTOR* 2012] performance is also given to show the **performance** differences between **OpenCL** and **CUDA** based solutions.
  - Note: **zero-copy** buffering scheme is adopted on the three **OpenCL** devices, and **one-copy** scheme is used on the **CUDA** device (CUDA/ GTX 1060 GPU)

Preliminary Results

- **Baseline** is **AES-NI**.
- **i7 multicore processor** achieves up to 5x speedup (size > 2MB).
- **Discrete GPU** get 3.5x performance speedup.
- **Integrate HD GPU** get 1.2x performance speedup.
- Thanks to zero-copy buffering **OpenCL** is better than **CUDA** version(size > 2MB).



Conclusion

- **Benefit:**
  - OS services would get performance boost with **KOCL**.
  - OS developers could focus more on the application logic while KOCL handles the hardware dependent details:
    - Zero-copy buffering scheme is offered to reduce the programming efforts and to facilitate computation acceleration.
    - **Developers** are allowed to choose different specific accelerator for computation.
- **Not covered by KOCL:**
  - KOCL simplifies the overhead for heterogeneous computing, but it still requires the developers to take care of parallel computing:
    - Judicious partitioning tasks for acceleration.
    - Tweaking the OpenCL parameters.

Thank you !

Question ?