

Transparent partial page migration between CPU and GPU

**Shiqing ZHANG, Zheng QIN, Yaohua YANG, Li SHEN,
Zhiying WANG**

Frontiers of Computer Science, DOI:10.1007/s11704-018-7386-4

Problems & Ideas

- Problems of Full Page Migration applying a large page size
 - large load-to-use interval for on-demand requests
 - bandwidth waste and subsequent requests stall
- Ideas: Partial Page Migration
 - migrate only the requested portion of a page on demand
 - two implementations: (a) single valid range, (b) multiple valid ranges

- ➊ Data request generated by GPU core.
- ➋ GMMU query the status and valid range of the page.*
- ➌ GMMU allocates Buffer entry.
- ➍ GMMU send migration request to CPU.
- ➎ Migration Controller migrate the requested part of the page.
- ➏ GMMU Modify the valid status and range of the page.
- ➐ GMMU notifies GPU core to directly access the page.

* In Step 2, if the page is valid in GPU or is partial valid in GPU and the requested range is within the valid range of the page, go to Step 6.

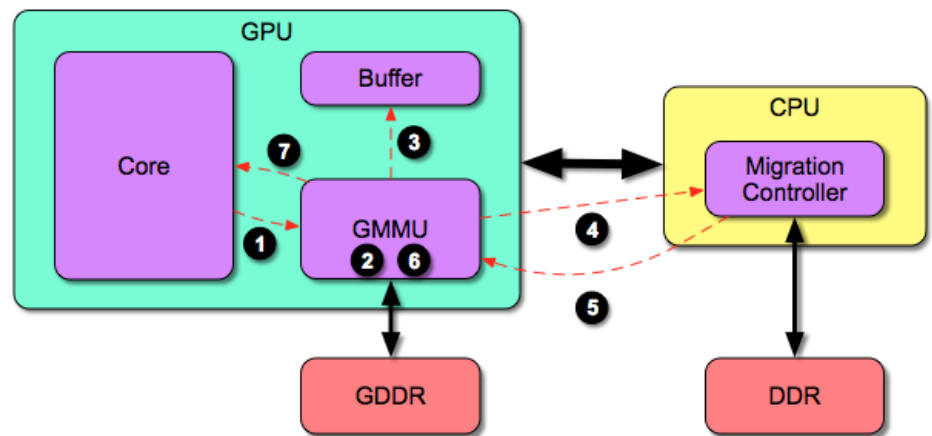


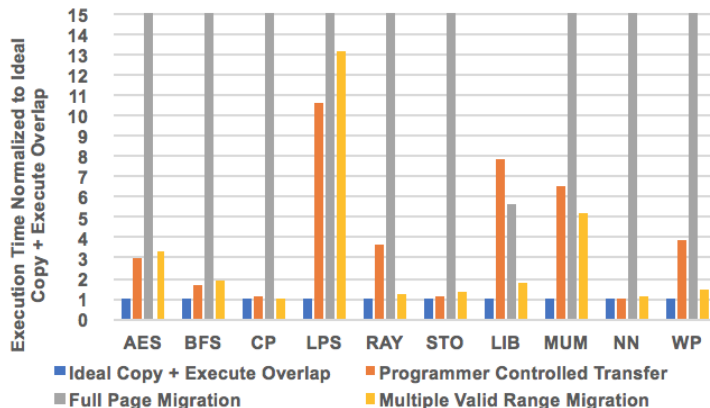
Figure 1. Architectural view of processing transparent partial page migration.

Main Contributions

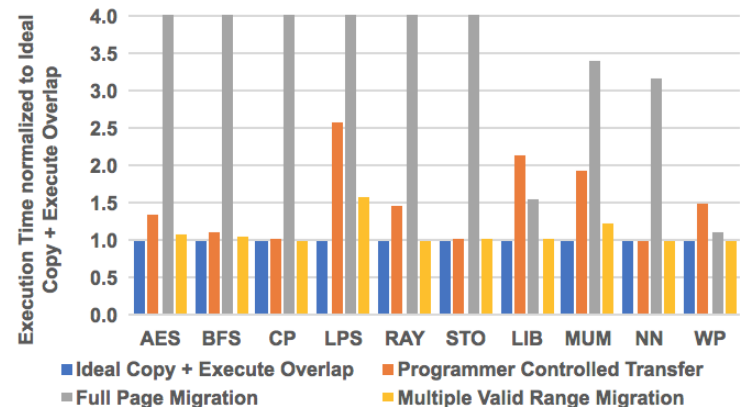
- **Contributions**

- propose a transparent partial page migration mechanism and two implementations: (a) single valid range, (b) multiple valid ranges
- evaluate the performance sensitivity of partial page migration to interconnection bandwidth and migration unit size
- experimental results show that partial page migration improved from full page migration's $72.72\times$ deceleration to $1.29\times$ acceleration compared to programmer controlled transfers under 16GB/sec bandwidth, and from $18.85\times$ deceleration to $1.37\times$ acceleration under 96GB/sec bandwidth.

- **Performance comparison**



(a) interconnection bandwidth = 16GB/sec



(b) interconnection bandwidth = 96GB/sec

Figure 2. Performance comparison.