DaeMon: Architectural Support for Efficient Data Movement in Fully Disaggregated Memory Systems

Christina Giannoula
Kailong Huang, Jonathan Tang, Nectarios Koziris, Georgios Goumas, Zeshan Chishti, Nandita Vijaykumar
Executive Summary

Problem:
Efficient data movement support is a major system challenge for fully Disaggregated Systems (DSs).

Contribution:
DaeMon: the first adaptive data movement solution for fully DSs.

Key Results:
DaeMon achieves 2.39x better performance and 3.06x lower data access costs over the widely-adopted scheme of moving data at page granularity.
What is resource disaggregation?
thanks to recent advances in network technologies
Benefits of Fully Disaggregated Systems

- Resource Utilization

Monolithic System

Disaggregated System
Benefits of Fully Disaggregated Systems

• Failure Handling
Benefits of Fully Disaggregated Systems

- Resource Scaling
Benefits of Fully Disaggregated Systems

- Heterogeneity

many different types of hardware devices over the network
Benefits of Fully Disaggregated Systems

- Resource Utilization
- Failure Handling
- Resource Scaling
- Heterogeneity

Disaggregated systems can **significantly decrease data center costs**
Baseline Disaggregated System

Network

Compute Component

CPU
Local Memory

Controller
Remote Memory

Memory Component

Memory Component

Memory Component

Memory Component
Baseline Disaggregated System

- Compute Component
- Compute Component
- Compute Component
- Compute Component
- Local Memory
- CPU
- Network
- Memory Component
- Memory Component
- Memory Component
- Memory Component
- Remote Memory

hosts ~20% of application’s data
Baseline Disaggregated System

Network

CPU
Local Memory

Compute Component

Compute Component

Compute Component

Remote Memory

Controller

hosts ~80% of application’s data

Memory Component

Memory Component

Memory Component

Memory Component
Baseline Disaggregated System

Data is typically moved at page granularity.
Baseline Disaggregated System

Network

- CPU
- Local Memory

- Remote Memory

- Controller

- Distributed OS modules

- Compute Component
- Memory Component

- Network
Why is data movement challenging?
#1: Coarse-Grained Data Migrations

- Page granularity (e.g., 4KB) data migrations:
  - Software transparency
  - Low metadata overheads
  - High spatial locality

Network CPU Local Memory

Controller
Remote Memory

high bandwidth consumption

latency-critical cache lines are slowed down
#1: Coarse-Grained Data Migrations

- Page granularity (e.g., 4KB) data migrations:
  - Software transparency
  - Low metadata overheads
  - High spatial locality

A latency-efficient and bandwidth-efficient solution is necessary
#2: Non-Conventional System Design

- Disaggregated systems are **not monolithic**

- Hybrid/heterogeneous memory systems:

  - Centralized memory management
  - Distributed memory management

  - Examples:
    - Thermostat [ASPLOS'17]
    - Kleio [HPDC'19]
    - Chameleon [MICRO'18]
    - HSCC [ICS'17]
    - Nimble [ASPLOS'19]
#2: Non-Conventional System Design

- Disaggregated systems are not monolithic
  - Hyphenated/heterogeneous memory systems:
    - CPU
    - Local Memory
    - Controller
    - Remote Memory
    - DRAM
    - Cache
    - …

  
- Hybrid/heterogeneous memory systems:
  - System-Level Solutions
  - Hardware-Level Solutions
  - centralized hardware units in the CPU side

  - would incur high hardware overheads

  - Chop [HPCA’10]
  - UH-MEM [CLUSTER’17]
  - MemPod [HPCA’17]
  - LGM [IPDPS’19] …
#2: Non-Conventional System Design

- Disaggregated systems are not monolithic

Prior solutions are not suitable or efficient for disaggregated memory systems
#3: Variability in Data Access Latencies

- Data access latencies depend:
  - **Location** of the remote memory component

![Diagram showing the relationship between CPU, local memory, remote memory, and data access latencies]

- Different locations for application’s data
#3: Variability in Data Access Latencies

- Data access latencies depend:
  - Location of the remote memory component
  - Network contention

[Diagram showing CPU, Local Memory, Controller, Remote Memory, and data with high contention due to concurrent jobs sharing the network]
#3: Variability in Data Access Latencies

A robust solution to variability in data access latencies is necessary.

Data placements can vary during runtime or between multiple executions.
How can we build an efficient solution?
1. Disaggregated Hardware Support

- Independence
- High Parallelism
- High Scalability

**Diagram:**
- Compute Component
  - CPU
  - LLC
  - FPGA
  - DaeMon Compute Engine
  - Local Memory
- Memory Component
  - Controller
  - DaeMon Memory Engine
  - Remote Memory

**Note:**
- Dedicated units

---

25
2. Multiple Granularity Data Movement

Compute Component

DaeMon Compute Engine

- CPU
- LLC
- Local Memory

- Sub-block Queue
- Page Queue

Memory Component

DaeMon Memory Engine

- Controller
- Remote Memory

- Sub-block Queue
- Page Queue
2. Multiple Granularity Data Movement

- **Compute Component**
  - DaeMon Compute Engine
  - CPU
  - LLC
  - Local Memory
  - Sub-block Queue
  - Queue Controller
  - Page Queue
  - Cache lines
  - Pages

- **Memory Component**
  - DaeMon Memory Engine
  - Controller
  - Remote Memory
  - Sub-block Queue
  - Queue Controller
  - Page Queue

**Prioritization of cache line migrations**
2. Multiple Granularity Data Movement

✓ Software Transparency
✓ Low Metadata Overheads
✓ High Spatial Locality
✓ Latency-Efficiency in Critical Data
3. Link Compression in Page Migrations

Compute Component
- DaeMon Compute Engine
  - CPU
  - LLC
  - Local Memory
  - Sub-block Queue
  - Page Queue
  - (De) Compr. Unit

Memory Component
- DaeMon Memory Engine
  - Controller
  - Remote Memory
  - Sub-block Queue
  - Page Queue
  - (De) Compr. Unit

Cache lines
- Compressed pages
- compressed pages inside the network
3. Link Compression in Page Migrations

- **Compute Component**
  - DaeMon Compute Engine
  - CPU
  - LLC
  - Sub-block Queue
  - Queue Controller
  - Page
  - Sub-block Queue
  - Queue Controller
  - Cache lines
  - Compressed
  - Page

- **Memory Component**
  - DaeMon Memory Engine
  - Controller
  - Sub-block Queue
  - Queue Controller
  - Page

- ✓ Bandwidth-Efficiency
- ✓ Critical Cache Line Prioritization
4. Selection Granularity Data Movement

Compute Component
- DaeMon Compute Engine
  - CPU
  - LLC
  - Local Memory
  - Sub-block Queue
  - Page Queue
  - Cache line, page or both?
  - (De) Compr. Unit

Memory Component
- DaeMon Memory Engine
  - Controller
  - Remote Memory
  - Sub-block Queue
  - Page Queue
  - (De) Compr. Unit

Cache lines and Pages

CPU
LLC
Local Memory
Remote Memory
4. Selection Granularity Data Movement

Compute Component
- DaeMon Compute Engine
  - CPU
  - LLC
  - Local Memory
  - Sub-block Queue
  - Page Queue
  - Inflight Sub-block and Page Buffers
  - (De) Compr. Unit
  - Sub-block
  - Page

Memory Component
- DaeMon Memory Engine
  - Controller
  - Remote Memory
  - Sub-block Queue
  - Page Queue
  - (De) Compr. Unit

track pending data migrations
4. Selection Granularity Data Movement

Compute Component
- DaeMon Compute Engine
  - CPU
  - LLC
  - Local Memory
  - Selection Granularity Unit
    - Sub-block Queue
    - Page Queue
    - Inflight Sub-block and Page Buffers
    - Queue Controller
    - (De) Compr. Unit

Memory Component
- DaeMon Memory Engine
  - Controller
  - Remote Memory
  - Sub-block Queue
  - Page Queue
  - Page Controller
  - (De) Compr. Unit

Cache lines, page or both?

33
4. Selection Granularity Data Movement

- Robustness
- Versatility
- Adaptivity to Runtime Changes
Why does this work?
Use Case 1: Memory Access Patterns

Compute Component

CPU

LLC

DaeMon Engine

Inflight Buffers

Selection Gran. Unit

Local Memory

Inflight Buffers Utilization

Sub-block

Page

Sub-block

Page

Sub-block

Page

Sub-block

Page

...
Use Case 1: Memory Access Patterns

Compute Component

- CPU
- LLC
- DaeMon Engine
- Inflight Buffers
- Selection Gran. Unit
- Local Memory

Memory Component

- Controller
- DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization

- Sub-block
- Page

Selection Gran. Unit

- Cache lines
- Compressed pages

Time

low locality within pages
Use Case 2: Network Characteristics

Compute Component

- CPU
- LLC
- DaeMon Engine
- Inflight Buffers
- Selection Gran. Unit
- Local Memory

Memory Component

- Controller
- DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization

- High bandwidth consumption

Time
Use Case 2: Network Characteristics

Compute Component
- CPU
- LLC
- DaeMon Engine
- Inflight Buffers
- Selection Gran. Unit
- Local Memory

Memory Component
- Controller
- DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization
- Sub-block
- Page

Time
- high bandwidth consumption
- low bandwidth consumption
Use Case 3: Data Compressibility

Compute Component
- CPU
- LLC
- DaeMon Engine
- Inflight Buffers
- Selection Gran. Unit
- Local Memory

Memory Component
- Controller
- DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization
- app1
- high data compressibility

Time
Use Case 3: Data Compressibility

Compute Component
- CPU
- LLC
- Inflight Buffers
- DaeMon Engine
- Selection Gran. Unit
- Local Memory

Memory Component
- Controller
- DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization
- app1: high data compressibility
- app2: low data compressibility

Cache lines
- Compressed pages

Gran. Unit
- Sub-block
- Page
Speedup in Real Applications

- Page
- ComprPage
- CacheLine
- CacheLine+Page
- DaeMon-Compr
- DaeMon

Workloads: kc, tr, pr, nw, bf, bc, ts, sp, sl, hp, pf, dr, rs, GM
Speedup in Real Applications

- Page
- ComprPage
- CacheLine
- CacheLine+Page
- DaeMon-Compr

Speedup

Workloads: kc, tr, pr, nw, bf, bc, ts, sp, sl, hp, pf, dr, rs, GM

14.6

1.95x
Speedup in Real Applications

![Graph showing speedup in real applications with different workloads and various techniques.

- Page
- ComprPage
- CacheLine
- CacheLine+Page
- DaeMon
- Compr

Workloads:
- kc
- tr
- pr
- nw
- bf
- bc
- ts
- sp
- sl
- hp
- pf
- dr
- rs
- GM

Speedup values include:
- 14.6
- 1.29x

44
Speedup in Real Applications

- Page
- ComprPage
- CacheLine
- CacheLine+Page
- DaeMon-Compr
- DaeMon

Workloads:
- kc
- tr
- pr
- nw
- bf
- bc
- ts
- sp
- sl
- hp
- pf
- dr
- rs
- GM

Speedup:
- low locality within pages
- 1.09x
- 0.95x

Page dimensions: 720.0x540.0
Speedup in Real Applications

The chart illustrates the speedup of various workloads under different cache line and page compression techniques. The x-axis represents different workloads, while the y-axis shows speedup. The chart compares different compression methods:
- Page
- ComprPage
- CacheLine
- CacheLine+Page
- DaeMon-Compr
- DaeMon

Some notable speedup values include:
- nw: 8.4
- 14.6
- 11.7

The chart shows that DaeMon-Compr provides a significant speedup across most workloads, with a notable 1.53x speedup for DaeMon.
Speedup in Real Applications

DaeMon performs **best** in real-world applications
Data Access Costs in Real Applications

- Page
- ComprPage
- DaeMon-Compr
- DaeMon

低本地性于页面内
Data Access Costs in Real Applications

- Page
- ComprPage
- DaeMon-Compr
- DaeMon

Data Access Costs

- medium locality within pages
Data Access Costs in Real Applications

- Page
- ComprPage
- DaeMon-Compr
- DaeMon

Data Access Costs vs. Applications:

- kc
- tr
- pr
- nw
- bf
- bc
- ts
- sp
- sl
- hp
- pf
- dr
- rs
- GM

High locality within pages

50
Data Access Costs in Real Applications

DaeMon significantly reduces data access costs in real-world applications
DaeMon constitutes a scalable solution.
Speedup in Multiple Co-Running Jobs

- **DaeMon** over Page

1.96x

![Bar chart showing speedup over page for various workloads.](image)

DaeMon constitutes a **versatile** solution.
Conclusion

- Data movement is a major challenge for fully DSs
- Prior solutions are not suitable or efficient
- DaeMon is the first adaptive data movement solution
- DaeMon consists of four techniques:
  - Disaggregated hardware support
  - Decoupled multiple granularity data movement
  - Link compression in page movements
  - Selection granularity data movement
- DaeMon’s benefits over the widely-adopted scheme:
  - 2.39x better performance
  - 3.06x lower data access
- DaeMon is highly-efficient, low-cost, scalable and robust
DaeMon: Architectural Support for Efficient Data Movement in Fully Disaggregated Memory Systems

Christina Giannoula
Kailong Huang, Jonathan Tang, Nectarios Koziris, Georgios Goumas, Zeshan Chishti, Nandita Vijaykumar

Thank you!

arxiv.org/abs/2301.00414