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?
Monolithic vs Disaggregated Systems

thanks to recent advances in network technologies
Benefits of Fully Disaggregated Systems

- Resource Utilization

Monolithic System

- fits a few jobs

Disaggregated System

- fits many jobs

Network

- idle

- idle
Benefits of Fully Disaggregated Systems

• Failure Handling

Monolithic System

Disaggregated System
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

Memory Component

Compute Component

Memory Component

Compute Component

Memory Component

Remote Memory

Controller

CPU

Local Memory
Baseline Disaggregated System

Compute Component

CPU

Local Memory

hosts ~20% of application’s data

Network

Compute Component

Remote Memory

Controller

Memory Component
Baseline Disaggregated System

- Compute Component
- Compute Component
- Compute Component
- CPU
- Local Memory
- Controller
- Remote Memory

Network

Memory Component
Memory Component
Memory Component
Remote Memory

hosts ~80% of application’s data

hosts ~80% of application’s data
Baseline Disaggregated System

Compute Component

Memory Component

Network

CPU

Local Memory

Controller

Remote Memory

data is typically moved at page granularity
Baseline Disaggregated System

Network

Compute Component

Compute Component

Compute Component

CPU

Local Memory

distributed OS modules

Memory Component

Memory Component

Memory Component

Memory Component

Memory Component

Controller

Remote Memory
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

![Diagram of network and memory components with labels: CPU, Local Memory, Remote Memory, Controller, and bandwidth consumption and latency-critical cache lines.]
#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:

  - System-Level Solutions
    - Thermostat [ASPLOS’17]
    - Kleio [HPDC’19]
    - Chameleon [MICRO’18]
    - HSCC [ICS’17]
    - Nimble [ASPLOS’19]

  - Centralized memory management
  - Distributed memory management
#2: Non-Conventional System Design

- Disaggregated systems are **not monolithic**

  ![Diagram]

  - **Centralized hardware units in the CPU side**

- Hybrid/heterogeneous memory systems:

  ![Diagram]

  - **System-Level Solutions**
    - Chop [HPCA’10]
    - UH-MEM [CLUSTER’17]
    - MemPod [HPCA’17]
    - LGM [IPDPS’19]...

  - **Hardware-Level Solutions**
#2: Non-Conventional System Design

- Disaggregated systems are **not monolithic**

- Hybrid/heterogeneous memory systems:

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 different locations for application’s data]
#3: Variability in Data Access Latencies

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

Memory Component

DaeMon Memory Engine

- Remote Memory
- Controller

Queue Controller
- Sub-block Queue
- Page Queue

Pages

Cache lines

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
  - Queue Controller
  - Page Queue
  - (De) Compr. Unit

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

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

- **Compute Component**: DaeMon Compute Engine
  - CPU
  - LLC
  - Sub-block Queue
  - Page
  - Queue Controller
  - Compressed Pages
- **Memory Component**: DaeMon Memory Engine
  - Sub-block Queue
  - Page
  - Queue Controller

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

Compute Component
- DaeMon Compute Engine
- CPU
- LLC
- Local Memory

Memory Component
- DaeMon Memory Engine
- Controller
- Remote Memory

Sub-block Queue
Page Queue
Queue Controller
Cache lines
Pages
(De) Compr. Unit

Cache line, page or both?
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

Queue Controller

Memory Component
DaeMon Memory Engine

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

Queue Controller

Sub-block
Page

Cache lines
Pages

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
    - (De) Compr. Unit

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

Cache lines
Pages
Cache line, page or both?

Sub-block Page
4. Selection Granularity Data Movement

✔ Robustness
✔ Versatility
✔ Adaptivity to Runtime Changes
Why does this work?

DaeMon
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

Time

Inflight Buffers Utilization
- Sub-block
- Page
- high locality within pages
Use Case 1: Memory Access Patterns

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

Memory Component
- Controller
- DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization
- Sub-block
- Page

Cache lines
- Compressed pages

Time
- low locality within pages
- 37
Use Case 2: Network Characteristics

Compute Component

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

Memory Component

- Controller
- DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization

- Sub-block
- Page

Time

38

High bandwidth consumption
Use Case 2: Network Characteristics

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

- **Memory Component**
  - Controller
  - DaeMon Memory Engine
  - Remote Memory

- **Inflight Buffers Utilization**
  - High bandwidth consumption
  - Low bandwidth consumption

- **Sub-blocks and Pages**
  - Inflight Buffers
  - Selection Gran. Unit
  - Compressed pages

- **Time**
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
- Sub-block
- Page

app1

high data compressibility

Time

Cache lines
Compressed pages
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
- app2: low data compressibility

Cache lines
Compressed pages
Speedup in Real Applications

![Speedup Chart](chart.png)
Speedup in Real Applications

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

Speedup for different workloads:
- kc: 14.6x
- tr: 1.29x
- pr: 5x
- nw: 4x
- bf: 3x
- bc: 2x
- ts: 1.5x
- sp: 1.2x
- sl: 1x
- hp: 1.2x
- pf: 1.1x
- dr: 1.1x
- rs: 1.1x
- GM: 1.29x
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:**
- 8.4x
- 14.6x
- 1.09x
- 0.95x

*low locality within pages*
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:
- 8.4
- 11.7
- 14.6

1.53x
DaeMon performs **best** in real-world applications.
Data Access Costs in Real Applications

- Page
- ComprPage
- DaeMon-Compr
- DaeMon

Data Access Costs

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

low locality within pages
Data Access Costs in Real Applications

- Page
- ComprPage
- DaeMon-Compr
- DaeMon

Data Access Costs

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

medium locality within pages
Data Access Costs in Real Applications

- Page
- ComprPage
- DaeMon-Compr
- DaeMon

High locality within pages
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

<table>
<thead>
<tr>
<th>Workloads</th>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>bf-dr-ts-nw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bf-dr-ts-sp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pr-dr-sp-nw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pr-nw-ts-sp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bc-dr-ts-sp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tr-dr-ts-sp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bc-nw-ts-sp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tr-nw-ts-sp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dr-ts-sp-nw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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!