Bandwidth-Aware Page Placement in NUMA Systems

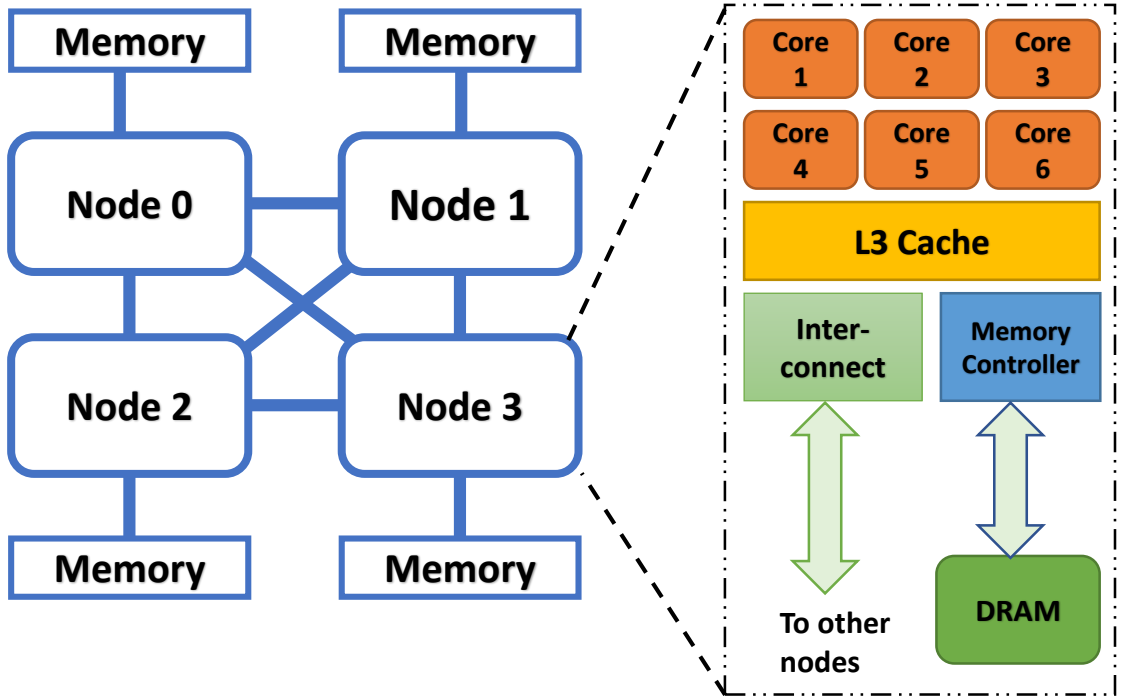
David Gureya[1,2], João Neto[1], Reza Karimi[3], João Barreto[1], Pramod Bhatotia[4], Vivien Quema[5], Rodrigo Rodrigues[1], Paolo Romano[1], and Vladimir Vlassov[2]

20th May 2020, IPDPS-2020

1 - INESC-ID, IST, University of Lisbon, Portugal | 2 - KTH Royal Institute of Technology, Sweden |
3 - Emory University, USA | 4 - University of Edinburgh, UK | 5 - Grenoble INP/ENSIMAG, France

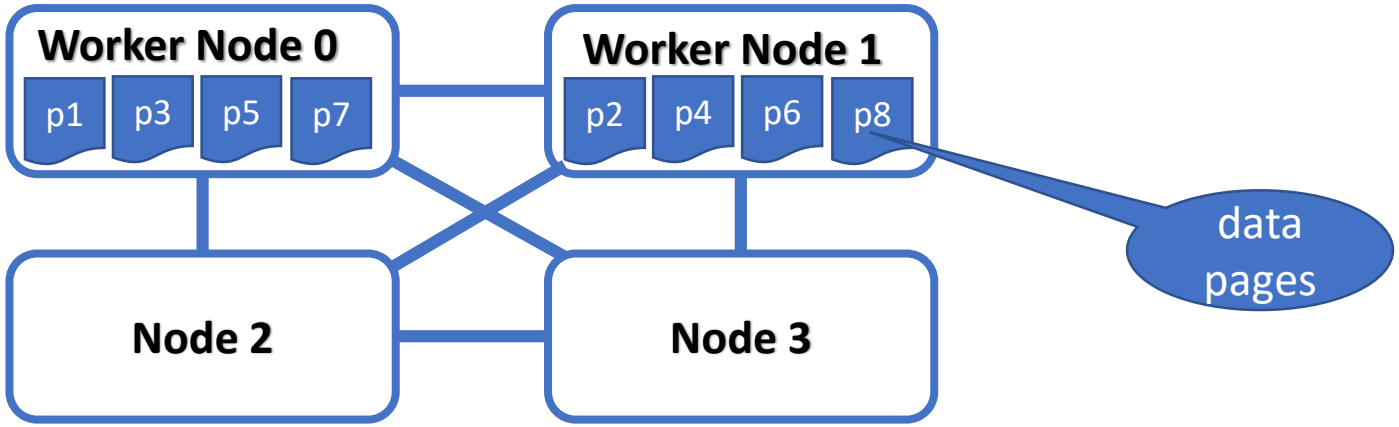
Page placement in Non-Uniform Memory Access (NUMA)

- NUMA emerging as the norm in high-end servers
- In a NUMA system, the access bandwidth/latency depends on where the data resides



Page placement problem in NUMA:
Where should the OS map the data pages of an application to optimize its performance?

“Uniform-workers” Placement Strategy



Two scenarios where application’s threads are clustered on a subset of nodes (worker nodes):

- When the application is deployed in a given node partition of a co-scheduled system
- When the application does not scale beyond a subset of the available cores

- If the application is bandwidth(BW)-intensive, uniform-workers is better than **“first-touch”**
- Therefore, it is used in most SoTA schemes

However, this strategy fails to **maximize memory throughput** in modern asymmetric NUMA systems by considerable margins:

- Distributes pages evenly, even though the BWs are asymmetric
- Does not use the available BW of the other nodes

BWAP: Bandwidth-Aware Page Placement

- **Goal:** Devise and enforce an efficient interleaving of the application's pages across NUMA nodes

- **Main focus:**
  - Bandwidth-intensive applications
  - Running in a subset of nodes of a larger NUMA system
    - co-scheduled scenario
    - scenario where applications do not scale

Key insights of BWAP

1. To maximize bandwidth, consider placing the application's pages on **every node**
   - Including nodes where the application is not running
2. To take bandwidth asymmetry into account, use **weighted interleaving**
   - Assign different weights to different nodes
   - Each node's weight denotes the fraction of pages that will be mapped to that node
3. Choose weights according to the **architecture** and the **application**, by combining:
   - Analytical performance model of the target NUMA architecture *(canonical tuner)*
   - Incremental page migration for application-specific tuning *(DWP tuner)*

Canonical Tuner

- Agnostic of the target application, runs offline
- Models the memory bandwidth of the NUMA topology
- Calculates the optimal weight distribution that maximizes the performance of a reference bandwidth-intensive application
- Output: **canonical weight distribution**

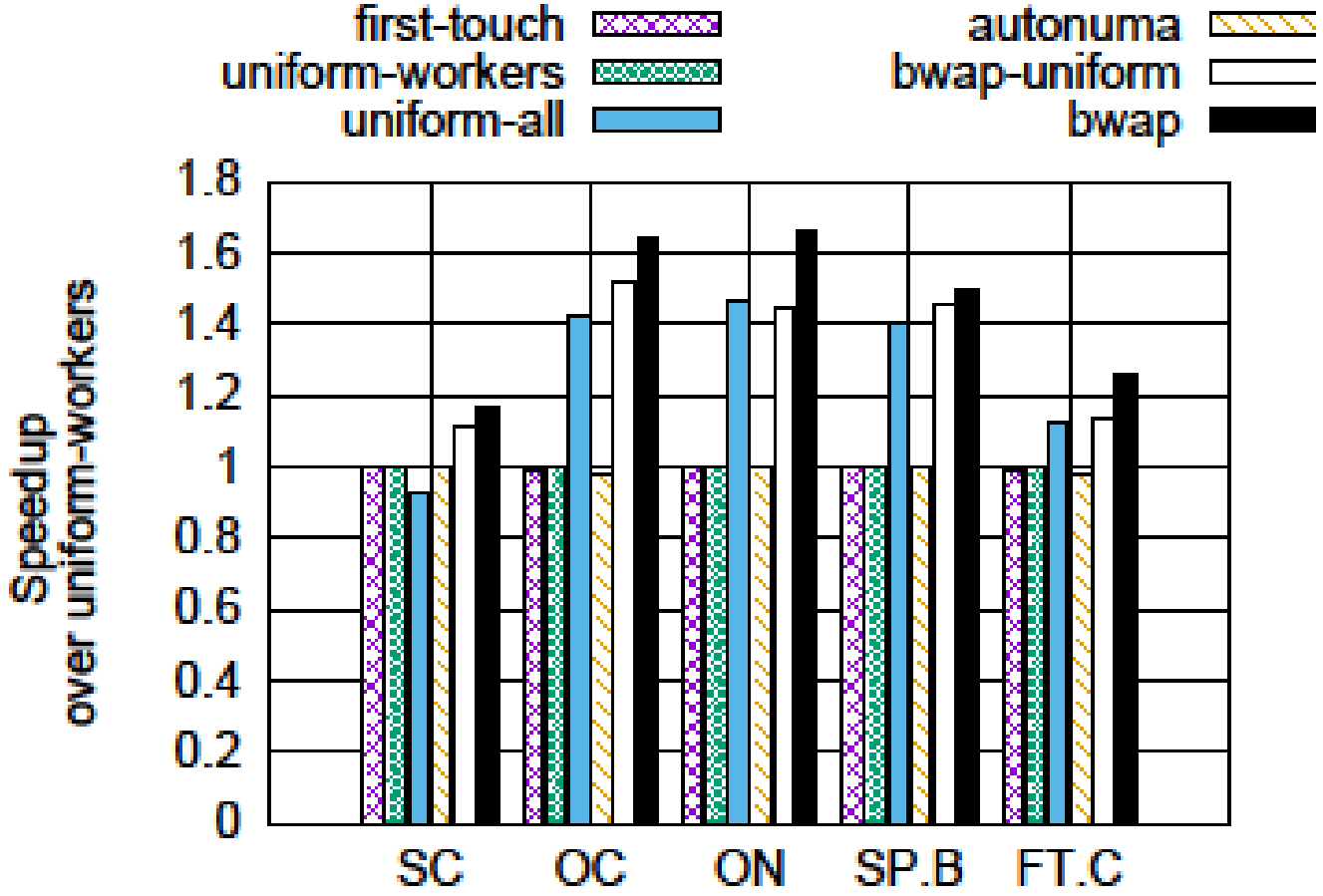
Data-to-Worker Proximity (DWP) Tuner

- Canonical weight distributions may not be suited for target application
- DWP tuner converts the canonical weight distribution to one that is optimized for the target application
- Finds an appropriate **data-to-worker proximity** factor (DWP)
  - DWP determines how many pages will be assigned to the set of worker nodes
  - Achieves this through an incremental page migration mechanism

Evaluation

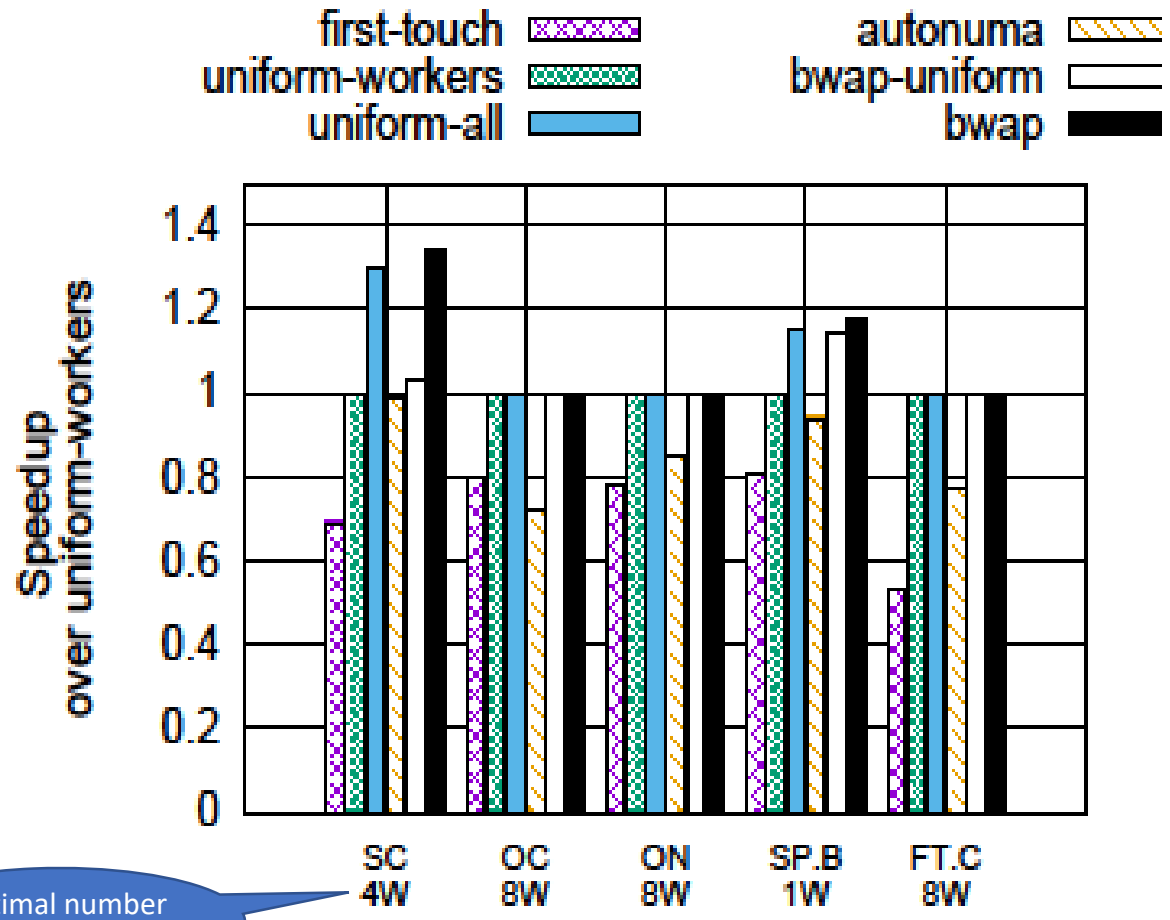
- Compare BWAP with state-of-the-art page placement policies
  - Linux default policy (first-touch)
  - Uniform-workers (uniform interleaving across workers)
  - Uniform-all (uniform interleaving across all nodes)
  - AutoNUMA
  - BWAP-uniform (variant of BWAP, which disables the canonical tuner)
- Benchmarks: multithreaded benchmarks from PARSEC, SPLASH, NAS
- Machines: 2 NUMA systems
  - AMD Opteron Processor, 8 NUMA nodes
  - Intel Xeon CPU E5-2660 v4, 4 NUMA nodes
- Execution scenarios: co-scheduled and stand-alone

Co-scheduled Scenario (1 worker node, 8-node machine)



- Best performing solutions are those that fully exploit the available memory BW by placing pages across all nodes, i.e., **uniform-all and BWAP**
- **BWAP** is able to outperform both **uniform-workers** and **autonuma** by up to 1.66*x*, and **uniform-all** by up to 1.50*x*

Stand-alone scenario (8-node machine)



- First-touch is usually the worst alternative for multi-worker scenarios
- Benefits of BWAP over the uniform interleaving alternatives drop when more workers are involved
- As applications use an increasingly larger worker node set:
  - The worker vs. non-worker dichotomy fades away.
  - The inter-worker canonical weight distributions tend to uniformity

Conclusions

- Today's usual techniques for page placement still rely on the obsolete assumption of a symmetric architecture
- We propose BWAP, a novel approach for asymmetric BW-aware placement of pages in NUMA systems
- BWAP improves the gains of state-of-the-art policies by up to 66%, on commodity NUMA machines
- BWAP is available at https://github.com/gureya/bwap

Corresponding author:
david.gureya@tecnico.ulisboa.pt

ACKNOWLEDGEMENTS :
This work has received funding from the European Union's Horizon 2020 research and innovation programme under the EPEEC project, grant agreement No 801051; from the FCT via the projects UIDB/50021/2020, and PTDC/EEI-SCR/1743/2014; and from the Erasmus Mundus Joint Doctorate in Distributed Computing (EMJD-DC) funded by the Education, Audiovisual and Culture Executive Agency of the European Commission under FPA 2012-0030.