EXTMEM: Enabling Application-Aware Virtual Memory Management for Data-Intensive Applications *[USENIX ATC '24]*

https://www.usenix.org/system/files/atc24-jalalian.pdf

The Artifact

Theodoros Moraitis, University of Athens

Artifact: **Available**

The artifact is available at: https://github.com/SepehrDV2/ExtMem

In the repo the below are included:

`src/` – *EXTMEM user-space library source code*

`linux/` – *Modified Linux kernel (based on v5.15)*

`run-scripts/` – *Scripts to reproduce experiments*

`swap-benchmarks/` – *Graph benchmarks (e.g., PageRank)*

`swap-microbenchmarks/` – *Microbenchmark tests*

Instructions on how to install and run it:

`README.md` – *Setup instructions and dependencies*

Artifact: **Functional**

Minimum System Requirements:

Given a dedicated swap file ExtMem can run on any multi-core X86 machine that can run the Linux 5.15>= kernel.

Recommended System Requirements:

For the best performance is recommended a dedicated NVMe SSD partition with async IO and at least 16 cores.

How I ran it:

1. QEMU/KVM Virtual Machine, given 2 Cores and 8GB of RAM
2. Ubuntu 22.04 → minimal install
3. Installed required dependencies
4. Cloned the EXTMEM repository
5. Initialized submodules
6. Built the custom Linux kernel
7. Booted into the new kernel
8. Ran EXTMEM benchmarks using provided scripts

Artifact: **Reproduced**

Recommended system requirements are not met on my current setup.

My options for reproduction were these two:
1) Cloud VM (e.g. AWS).
2) Modify benchmarks to fit current setup.

So, for the artifact results reproduction I chose option ***2) Modify/adjust benchmarks to fit my current low-end setup***:

- *QEMU/KVM Virtual Machine, given 2 Cores, 8GB of RAM, 120GB of SSD.*
- *Ubuntu 22.04 → minimal install*
- *Modified Linux 5.15*
- *Dedicated swap space of 8GB*

mmap microbenchmark	Paper	My VM setup
CPU Sockets	2	1
CPU Cores per socket	16 cores	2 cores
RAM	198GB DDR4	8GB DDR3
Working Set	16GB	8GB
DRAM Limits	8GB	4GB
Swap Space	NVMe dedicated	8GB swapfile

Steps:

- Build ExtMem (ExtMem/run-scripts/build-extmem.sh):
  ```
  cd /src
  make default
  make libextmem-pagerank.so  # not needed for now
  ```
- Modify mmapbench.cpp microbenchmark to allocate 8GB instead of 16GB:
  Change this: `uint64_t fileSize = 16ull * 1024 * 1024 * 1024;`
  To this: `uint64_t fileSize = 8ull * 1024 * 1024 * 1024;`
- Build microbenchmark (ExtMem/run-scripts/build-microbenchmarks.sh):
  ```
  cd swap-microbenchmarks/mmapbench
  g++ -O3 -g mmapbench.cpp -o mmapbench -ltbb -pthread
  ```
- Create the dedicated swapfile as a block (loop) device for ExtMem usage:
  ```
  sudo fallocate -l 8G /extmem_swapfile
  sudo chmod 600 /extmem_swapfile
  sudo losetup -f --show /extmem_swapfile   # /dev/loop18
  ```
- Run ExtMem-enabled swap-microbenchmark (mmapbench) giving it 4GB of DRAM limit before swapping:
  ```
  sudo DRAMSIZE=4294967296 SWAPDIR=/dev/loop18
  LD_PRELOAD=/home/teomorvm/Desktop/ExtMem/src/libextmem-
  default.so timeout 120s
  /home/teomorvm/Desktop/ExtMem/swap-microbenchmarks/mmap
  bench/mmapbench /dev/null 1 1 0 0 1 >
  /home/teomorvm/Desktop/ExtMem/my-results/extmem.log
  ```
- Create a common swapfile for fair Linux comparison:
  ```
  sudo fallocate -l 8G /swapfile
  sudo chmod 600 /swapfile
  sudo mkswap /swapfile
  sudo swapon /swapfile
  ```
- Run Baseline-Linux swap-microbenchmark (mmapbench) limiting it to 4GB of DRAM before swapping:
  ```
  sudo systemd-run --scope -p MemoryMax=4G timeout 120s
  /home/teomorvm/Desktop/ExtMem/swap-microbenchmarks/mmap
  bench/mmapbench /dev/null 1 1 0 0 1 >
  /home/teomorvm/Desktop/ExtMem/my-results/linux.log
  ```

In the two previous swap-microbenchmark executions (ExtMem and Linux) some manual parameters were given to mmapbench, more specifically `/dev/null 1 1 0 0 1`. These represent the below:

1. **dev**: Device Path - not used here, so /dev/null
2. **threads**: Number of Worker Threads
3. **seq(0)/rand(1)/hotset(2)**: Access Mode
4. **hint**: madvise Hint - 0: no madvice hint
5. **hugepage**: Huge Page Usage - 0: normal, 1: hugepages
6. **write**: Perform Writes - 0: reads only, 1: writes too

In order to test a wide combination of parameters, I created a script, which also automates the testing: `run-tests.sh`

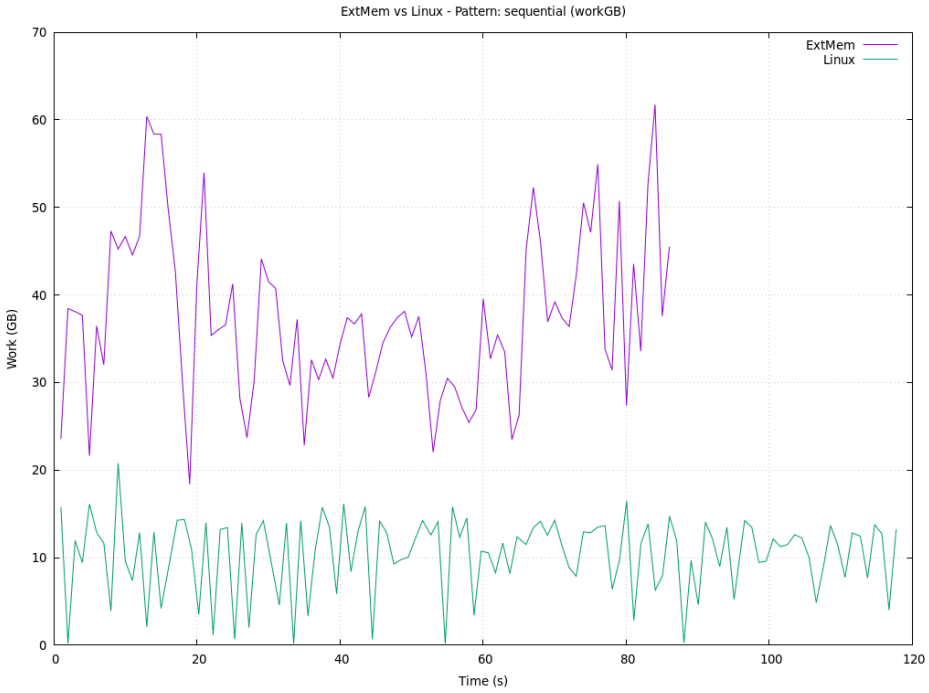
- After running it, these result-log files are created:

extmem_hugepage.log	linux_hugepage.log
extmem_normal.log	linux_normal.log
extmem_pattern_hotset.log	linux_pattern_hotset.log
extmem_pattern_random.log	linux_pattern_random.log
extmem_pattern_sequential.log	linux_pattern_sequential.log
extmem_read.log	linux_read.log
extmem_stress_hotset.log	linux_stress_hotset.log
extmem_stress_threads.log	linux_stress_threads.log
extmem_threads_1.log	linux_threads_1.log
extmem_threads_2.log	linux_threads_2.log
extmem_threads_4.log	linux_threads_4.log
extmem_write.log	linux_write.log

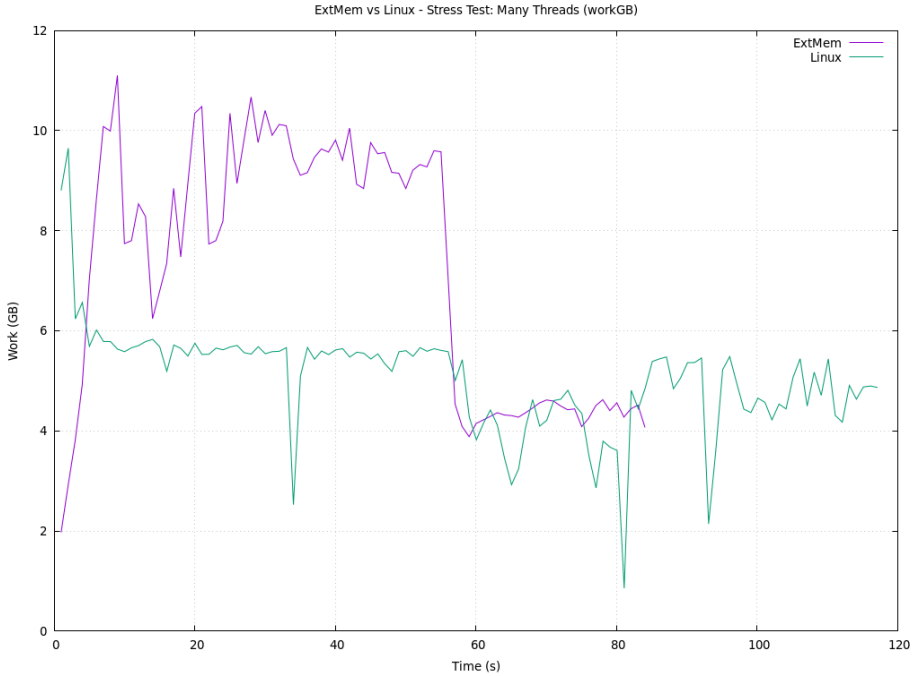
- Each row in any log has these information:
  `dev,pattern,hint,threads,time,workGB,tlb,readGB,CPUwork, Updates`
  For example, a row from `extmem_normal.log`:
  `/dev/null,1,0,4,1.0004,0.328751,20065,0,45,352993280`

- So, with all this data I created some plots, comparing the ExtMem vs Linux executions of mmapbench. For example:

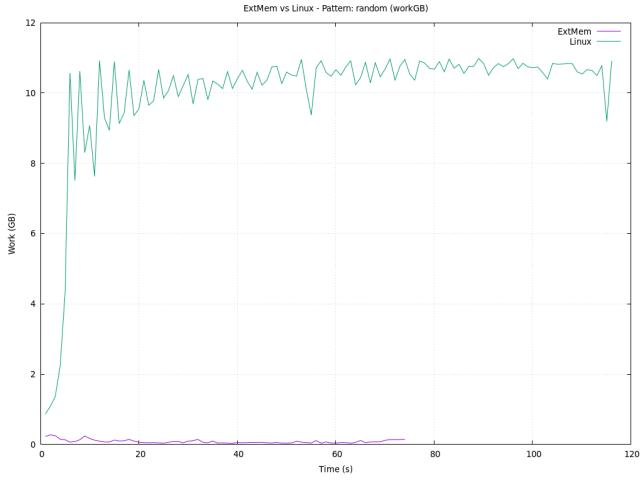
- `extmem_pattern_sequential` vs `linux_pattern_sequential`:

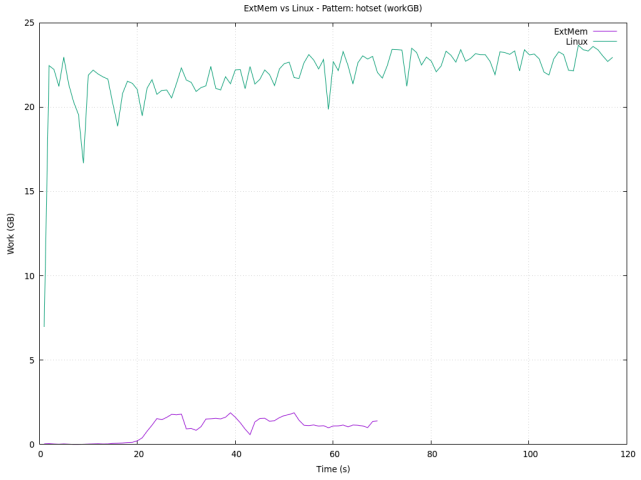


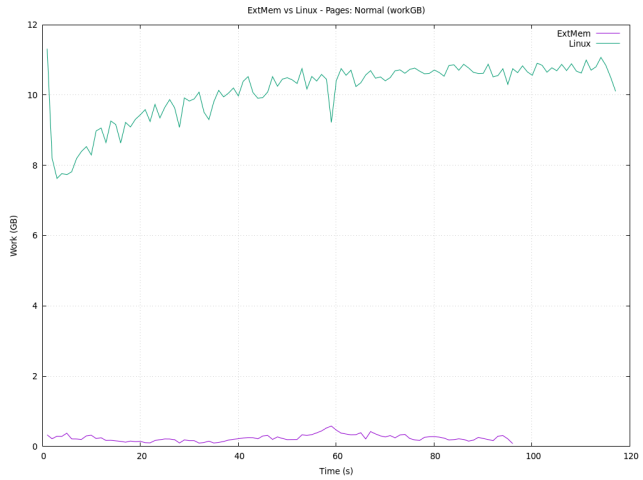
- `extmem_stress_threads` vs `linux_stress_threads`:

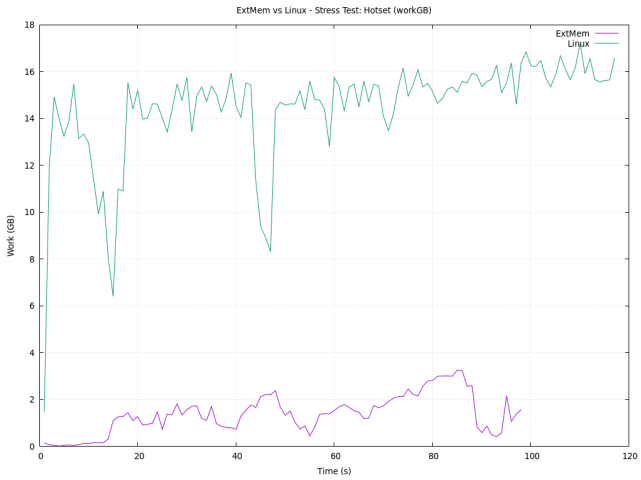


- Although ExtMem outperforms Linux in some cases like above, Linux delivers better performance in most tests overall, e.g.:









- The poorer performance of ExtMem compared to pure Linux in most cases may be due to configuration or execution issues, overhead introduced by ExtMem itself, or limitations of the virtual machine environment. Further investigation is needed to better understand these results.

- I only ran the mmap microbenchmark and did not include the PageRank test from the GAPBS suite, as the dataset and input sizes were too large for my limited VM setup.

- Plan to also run XSBench / MyMicrobenchmark, from the experimental handout, using ExtMem.