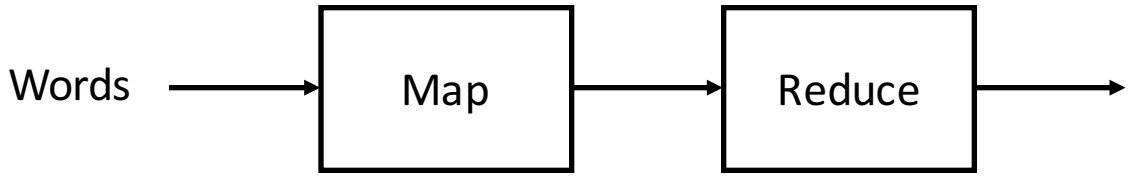
MR-MPI Project

Platform

- Hardware (Comet)
  - Number of nodes: 72 of 1944 available
  - Cores per node: 24
  - Memory per node: 128GB
  - Local scratch memory: 320GB SSD
- Software: MPICH 3.2

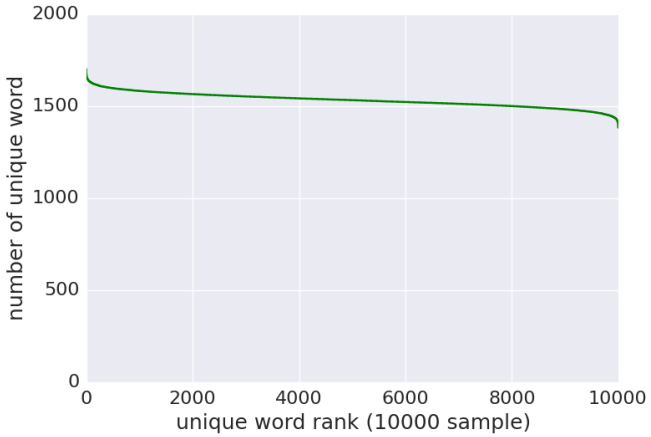
Benchmark 1: Wordcount

Workflow



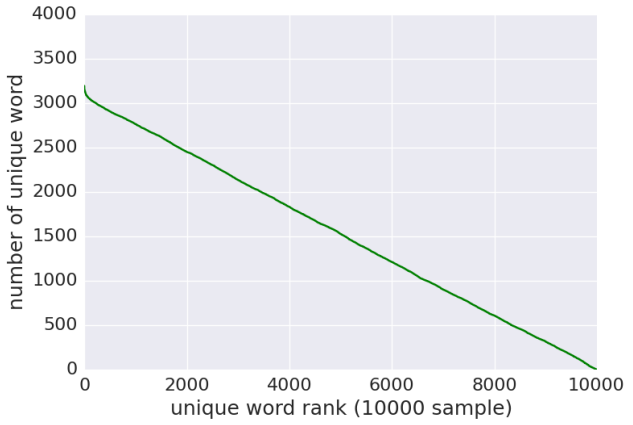
Dataset

Uniform Dataset



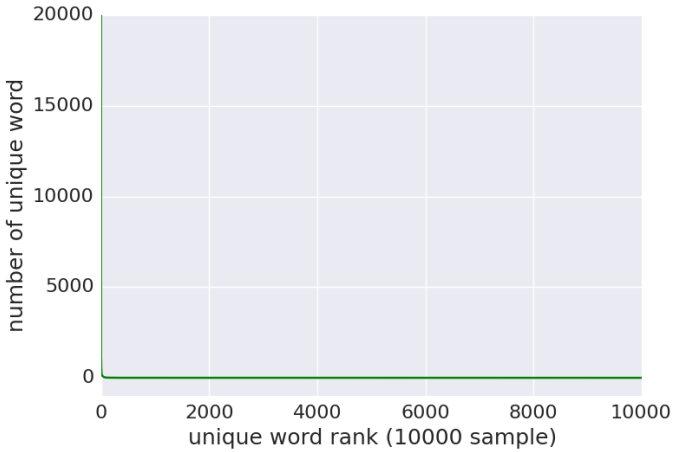
~100 000 unique words

Nonuniform Dataset



~100 000 unique words

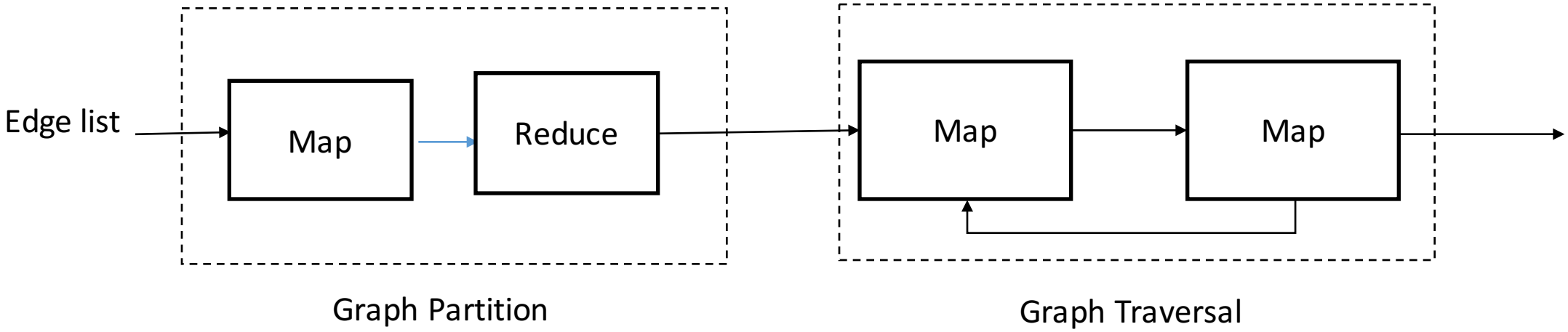
Wikipedia Dataset



> 2 000 000 unique words

Benchmark 2: BFS

Workflow

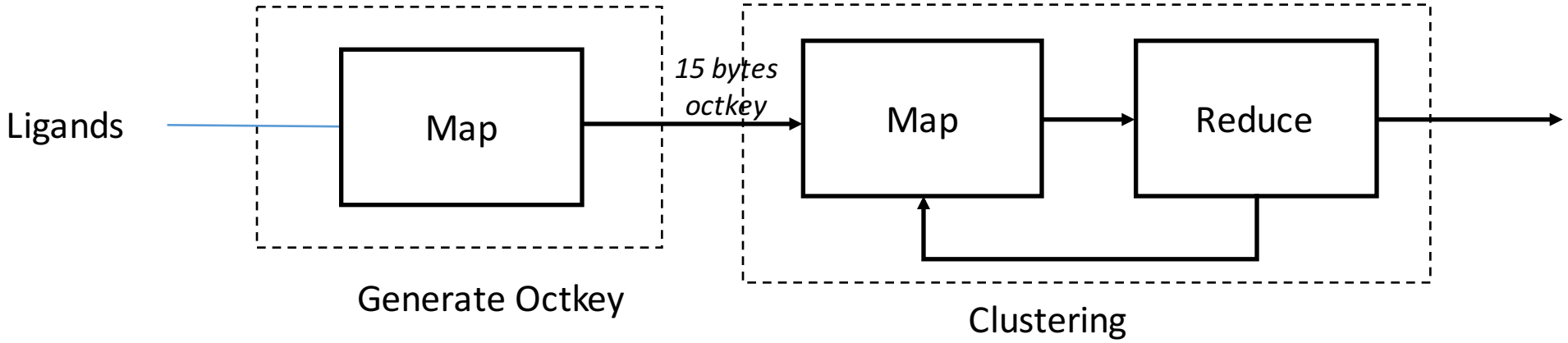


Dataset Scale-free graphs: generated by graph500 benchmark, average degree 32

Input file size	Vertexes
~8G	2^24
~16G	2^25
~32G	2^26
~64G	2^27=134 217 728

Benchmark 3: Octree clustering

Workflow

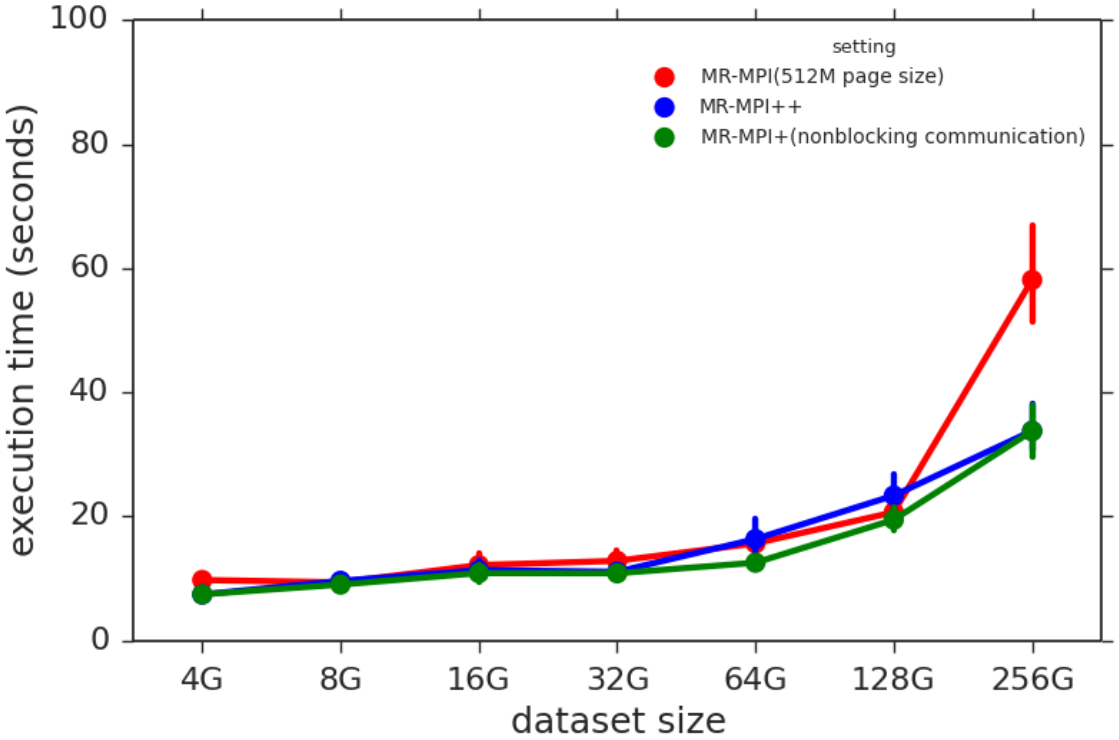


Dataset

1S: represent points of ligands follow the normal distribution, and the standard deviation=0.5

Input file Size	**Ligands**
128G	57,999,360
256G	115,998,720
512G	231,997,440
1T	463,994,880

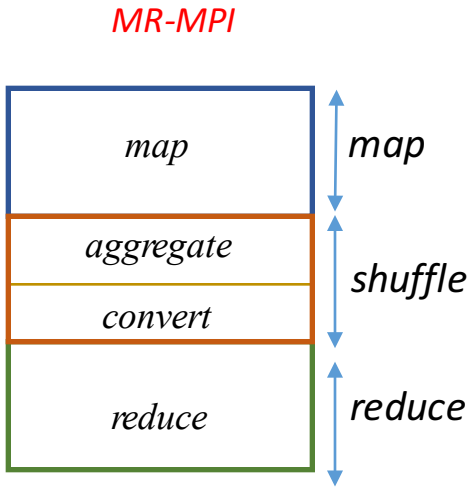
Week scalabiltiy



4G/node

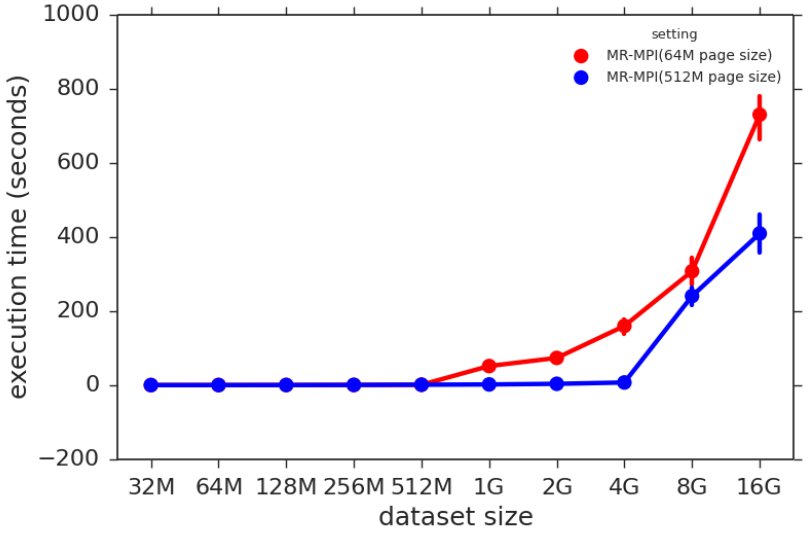
MapReduce-MPI Overview

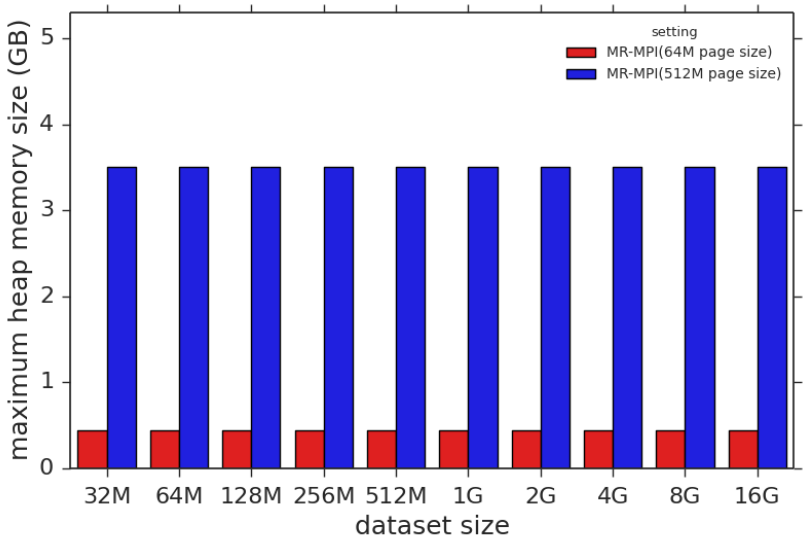
- MapReduce Phases
  - MR-MPI library can be divided into *map*, shuffle (*aggregate* and *convert* functions), *reduce* phase
- Data Structure
  - The library operates on two basic data structures KeyValue object (KV) and KeyMultiValue object (KMV).
  - The library handles the data in a predefined page size.
  - Minimum page count requirement in different functions: map(1), ***aggregate(7)***, convert(4+),reduce(3)
- Out-memory processing
  - If the data owned by a process in its collection of KeyValue or KeyMultiValue pairs fits within one page, then no disk I/O is performed.
  - If data exceeds the page size, then the data is written to temporary disk files and read back in for subsequent operations.



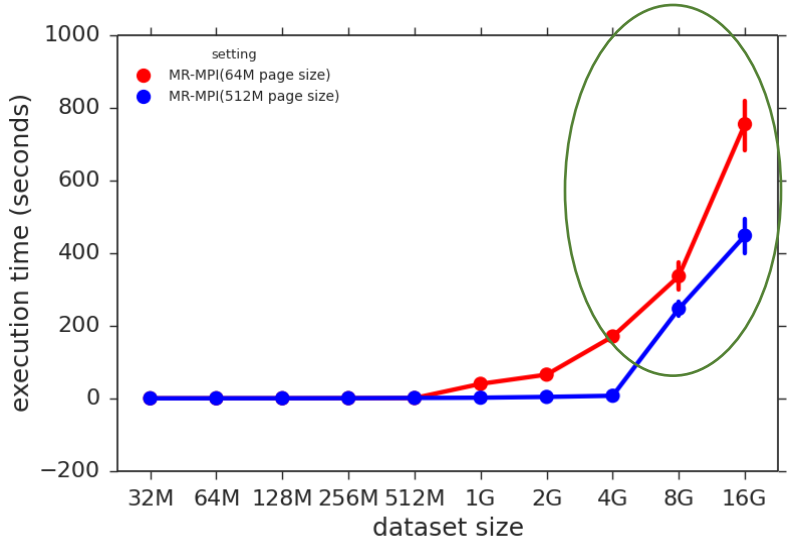
Wordcount Benchmark

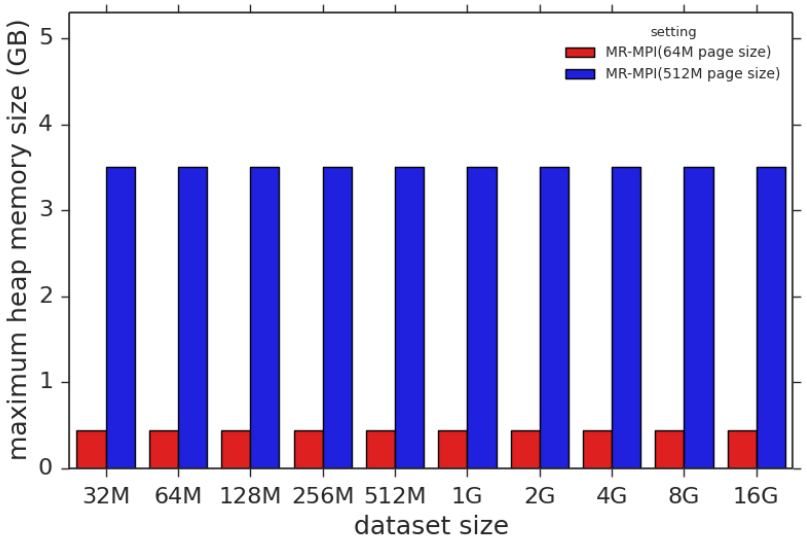
Uniform



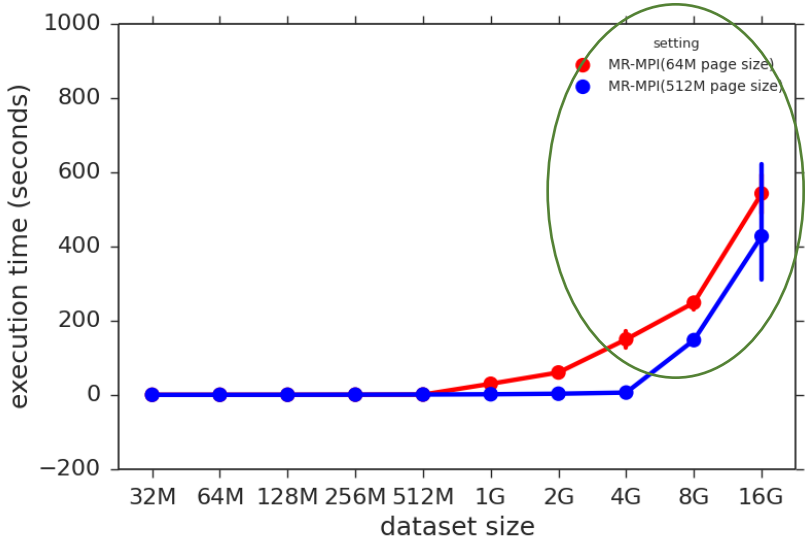


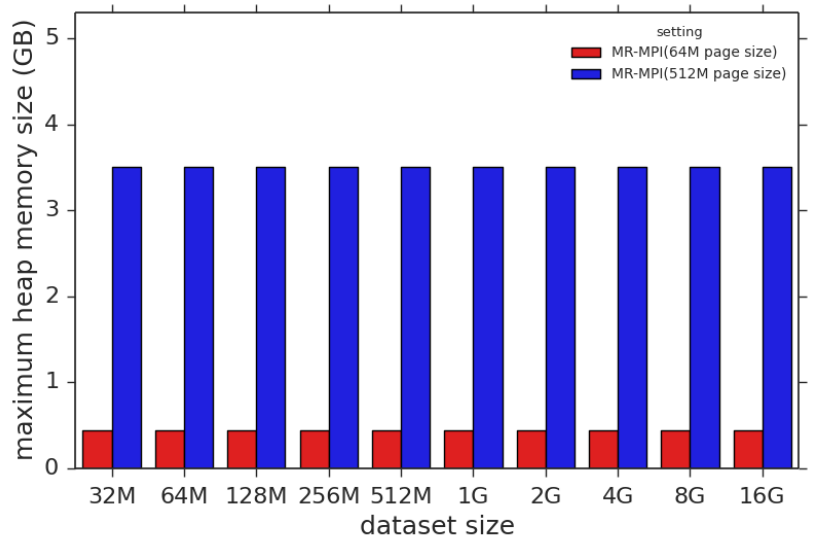
Nonuniform





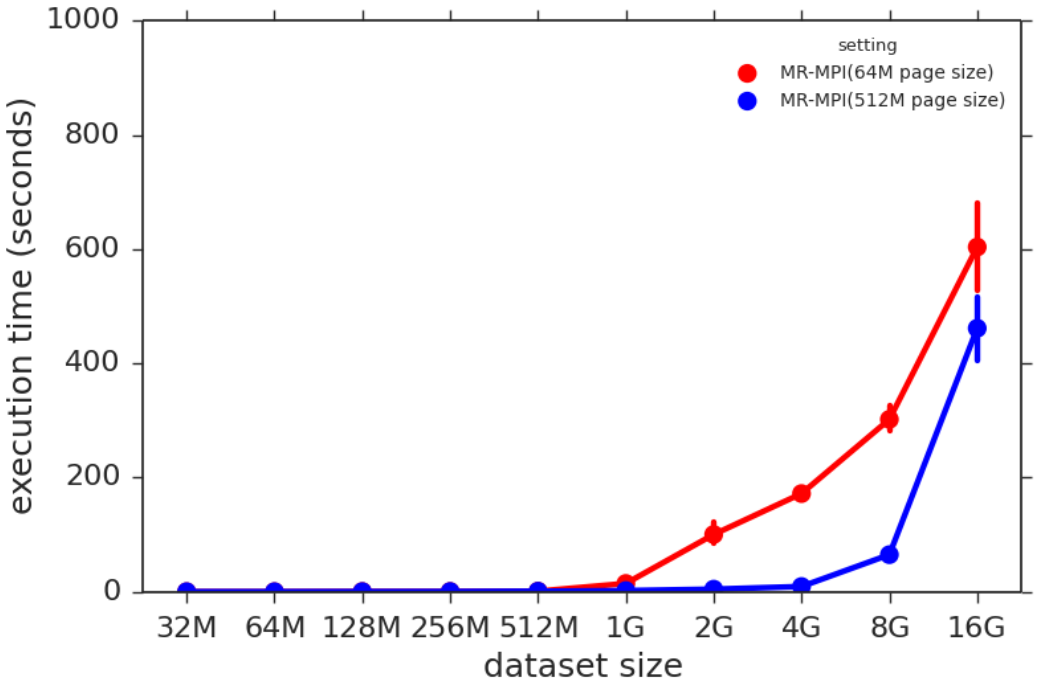
Wikipedia *Out-memory*

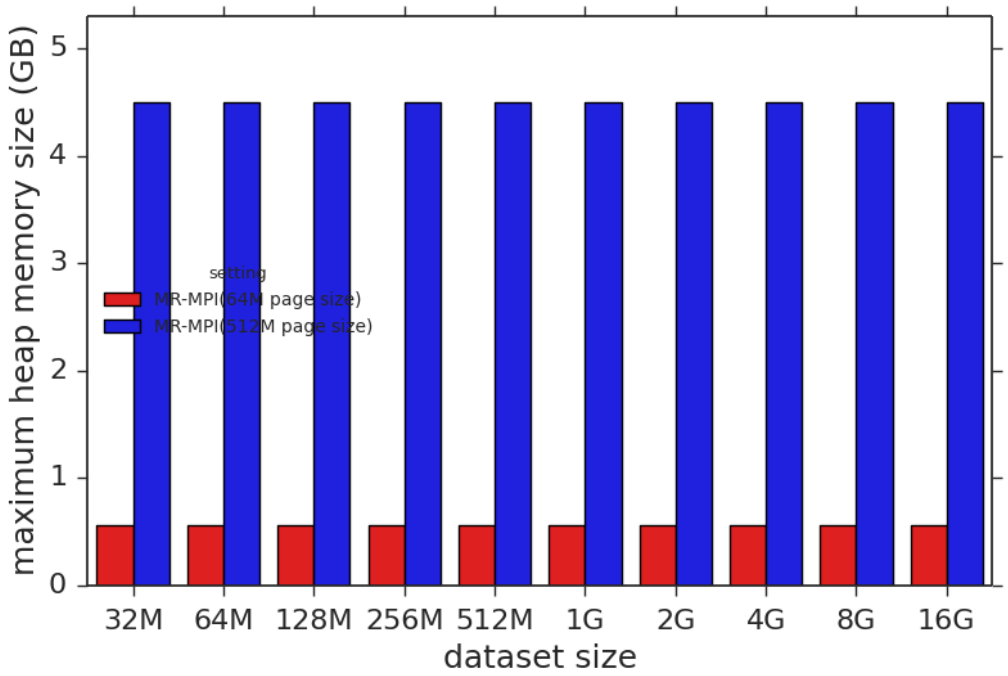




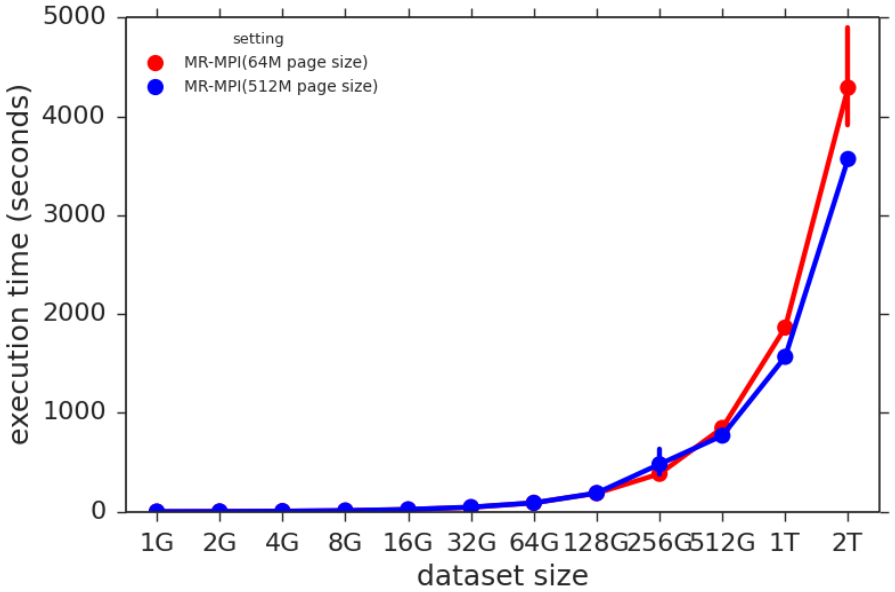
As out-memory performance is much worse than in-memory, we focus on in-memory only.

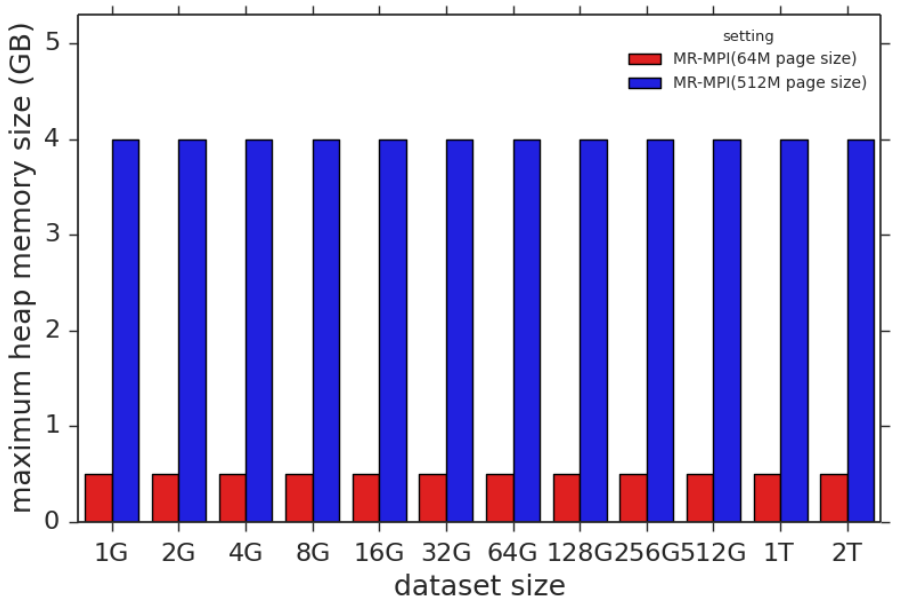
BFS


execution time (seconds)
1000
800
600
400
200
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
32M
64M
128M
256M
512M
1G
2G
4G
8G
16G
dataset size

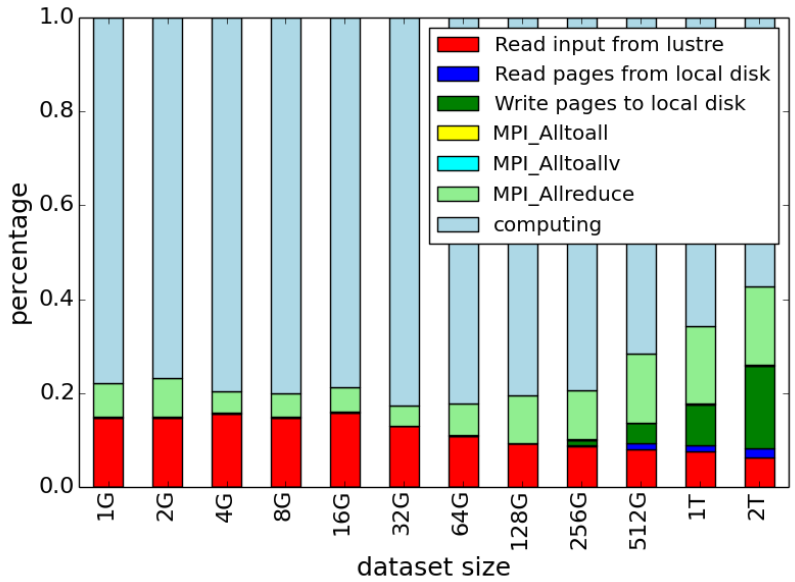

maximum heap memory size (GB)
5
4
3
2
1
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
32M
64M
128M
256M
512M
1G
2G
4G
8G
16G
dataset size

Octree Benchmark

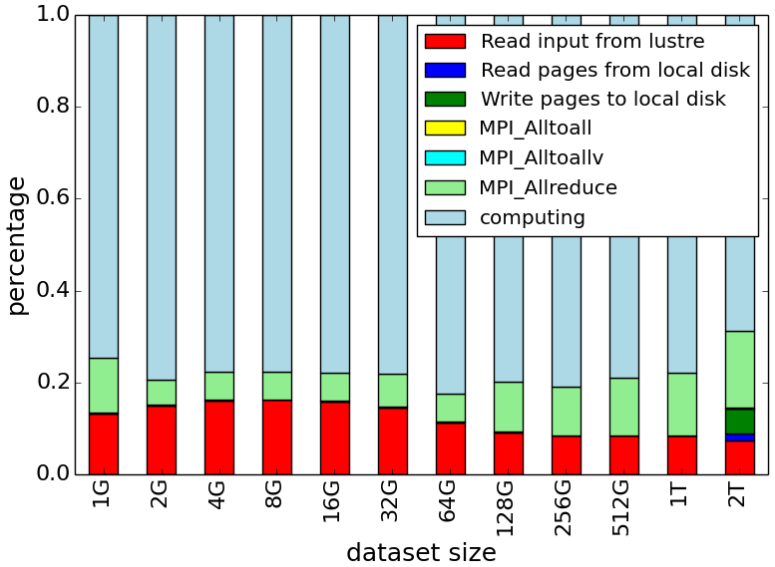




64M page size

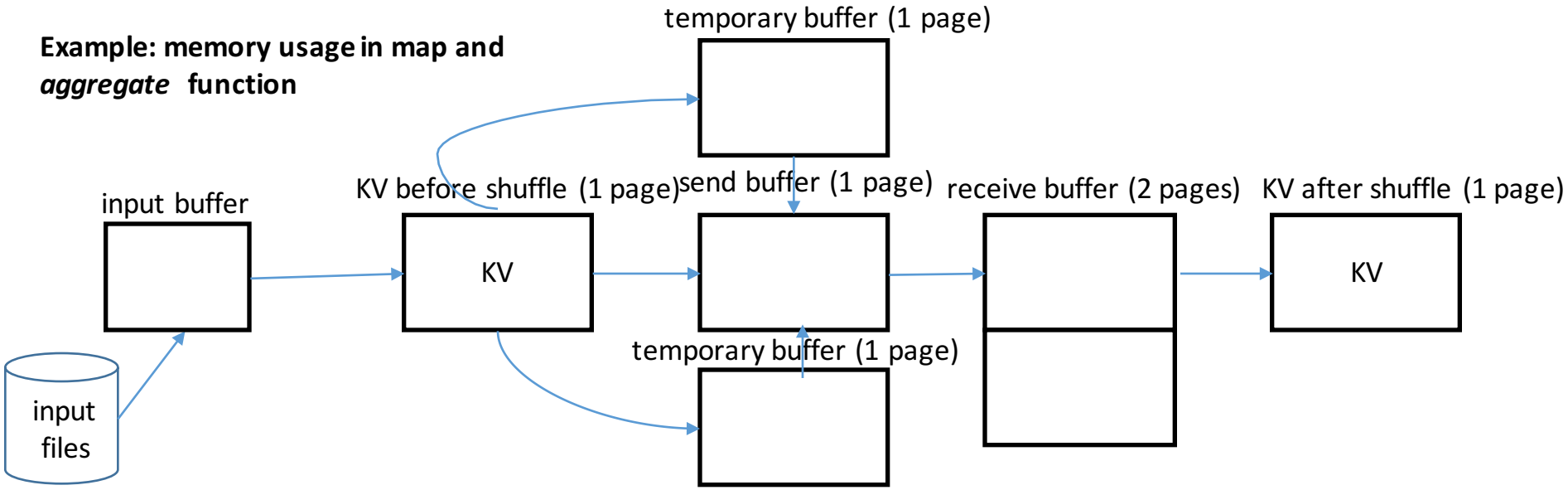


512M page size

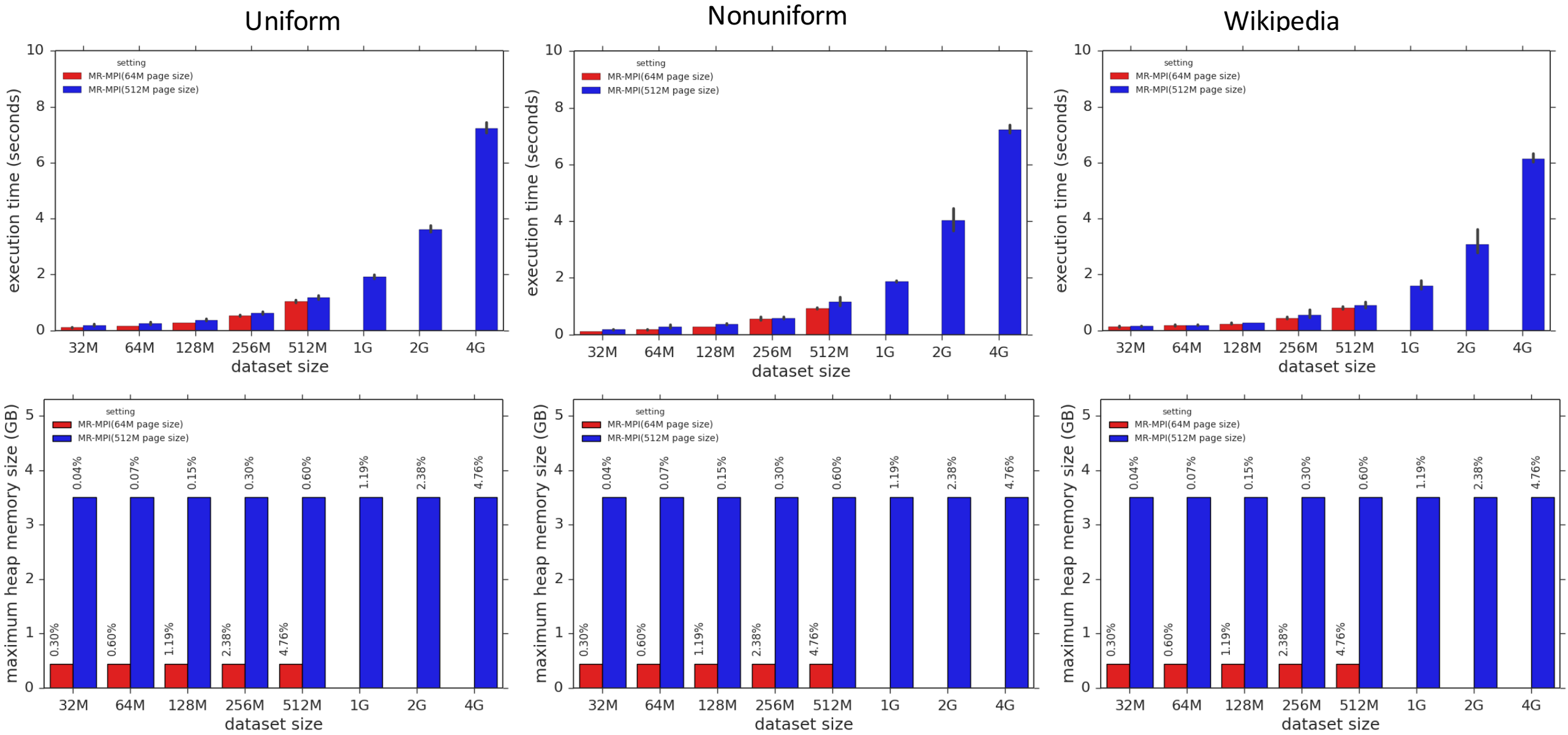


MR-MPI: Shortcoming of Memory Usage

- To ensure in-memory processing, the (KV/KMV) data owned by one process must fit in one page;
- Allocate most buffers based on the same page size, i.e communication buffers, temporary buffers.



Wordcount Benchmark

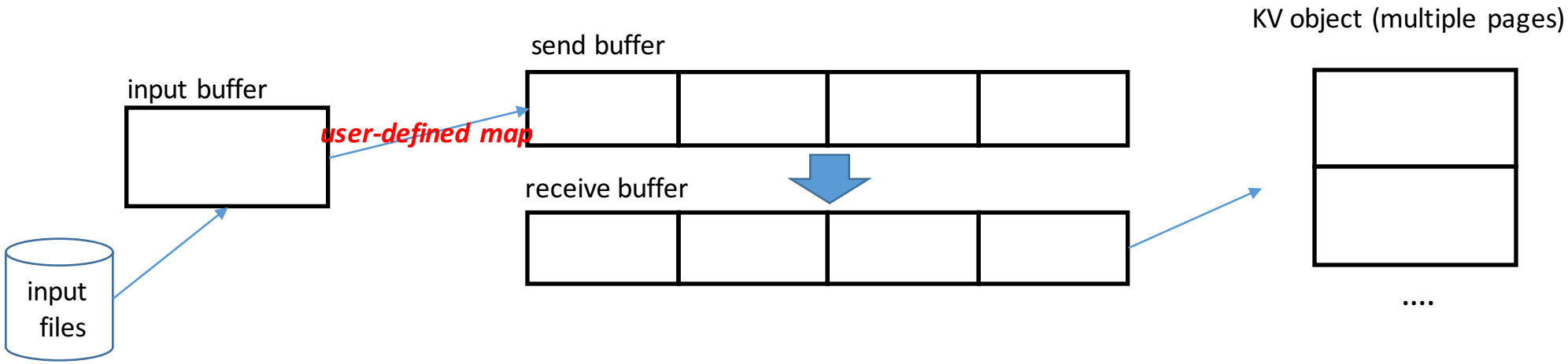


Memory usage efficiency is calculated by dataset size/maximum heap memory size (at top of the bar).

MR-MPI++: Improve memory usage efficiency

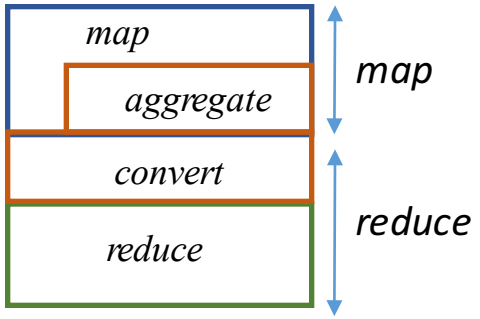
- Basic Ideas
  - Each process can have multiple pages to hold the (KV/KMV) data.
  - Allocate different buffers (input buffer, communication buffer and so on)based on different parameters; users can tune the best parameters for it.
  - Put results KV of map into communication buffer directly and reduce buffers needed in the communication phase.

Buffer structures in *map* phase and *aggregate* phases.

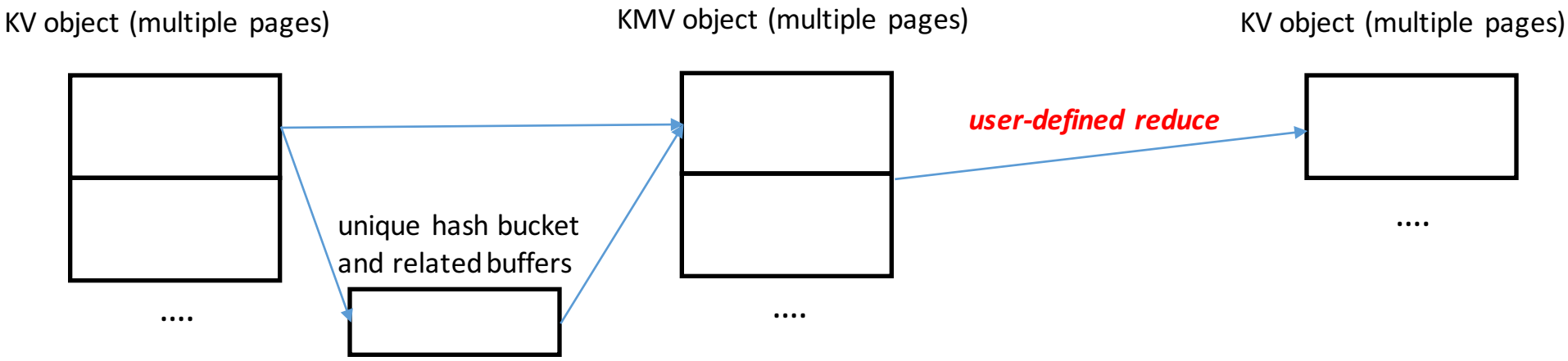


MR-MPI++: Improve memory usage efficiency

- Main Buffer Parameters
  - Page size: buffer size used to hold KV/KMV data
  - Input buffer size: buffer size for reading input files
  - Communication buffer size: communication buffer size
  - Unique bucket size: hash bucket for unique words



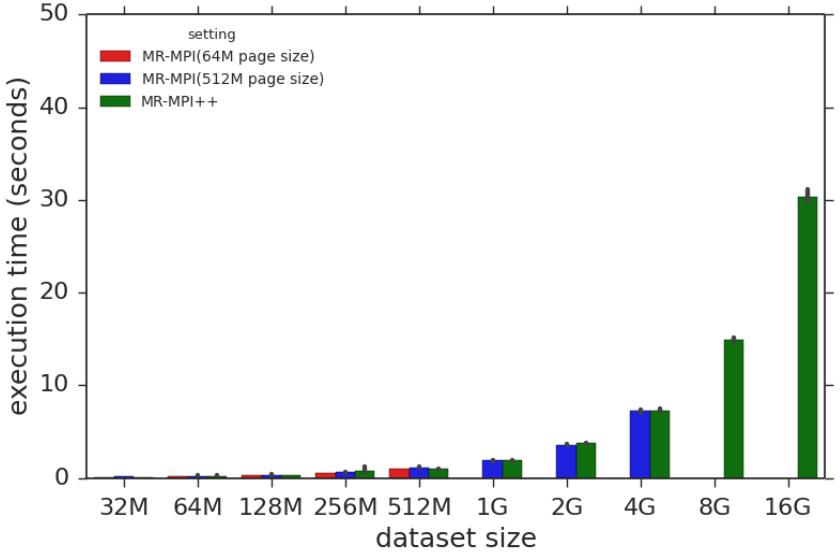
Buffer structures in *convert* phase and *reduce* phases.

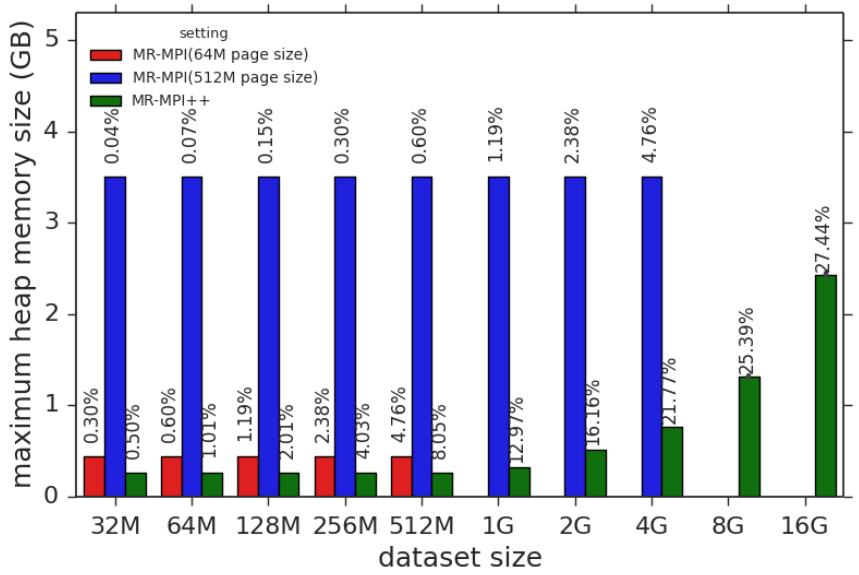


MR-MPI configuration:
page size: 64M
communication buffer size: 64M
input buffer size: 512M
unique bucket size: 2^17

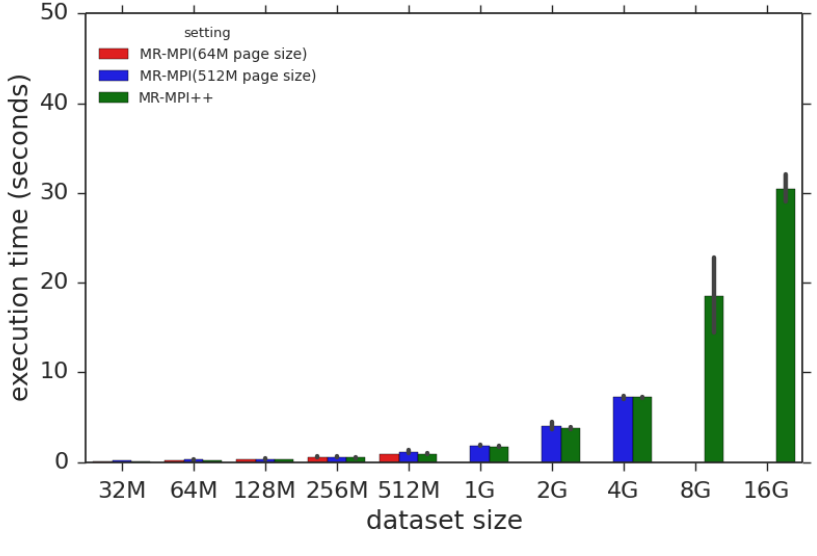
Wordcount Benchmark

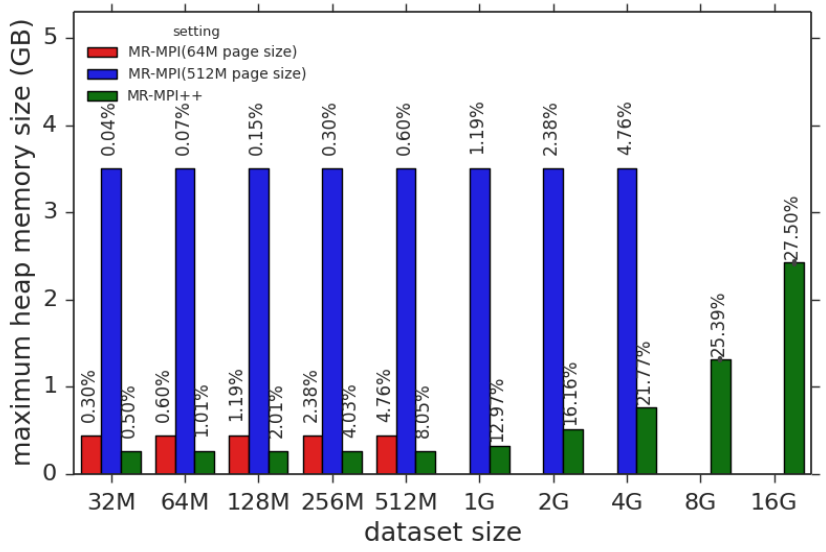
Uniform



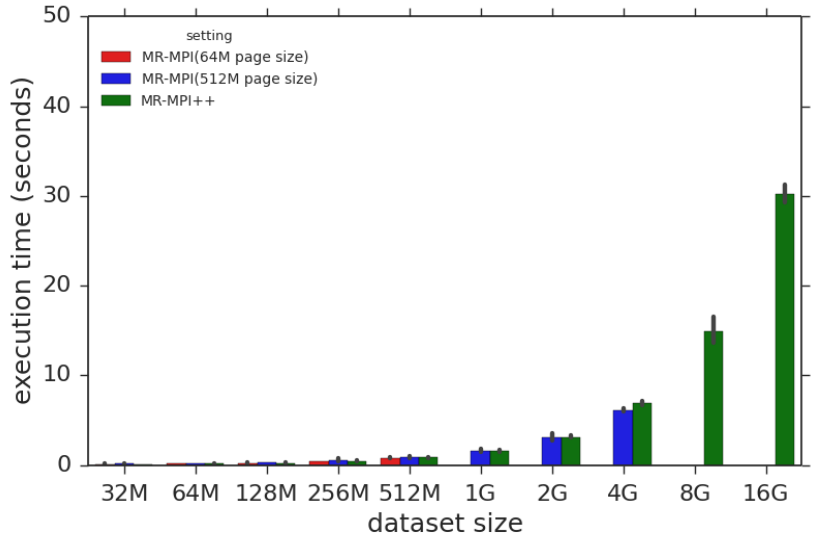


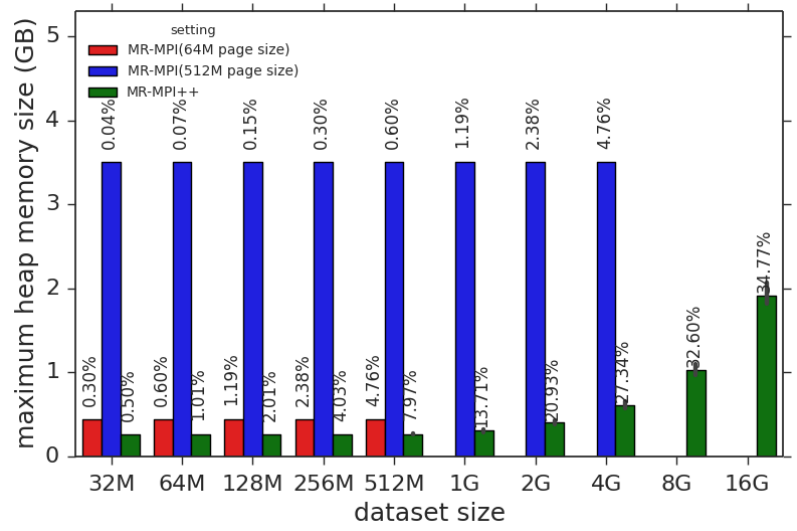
Nonuniform





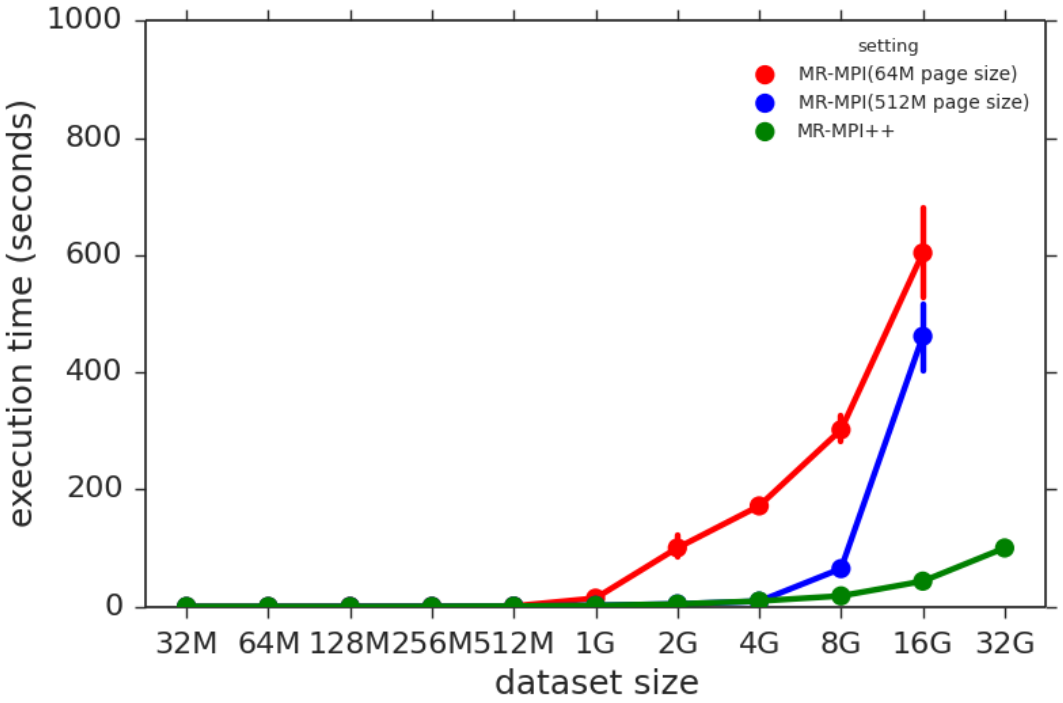
Wikipedia

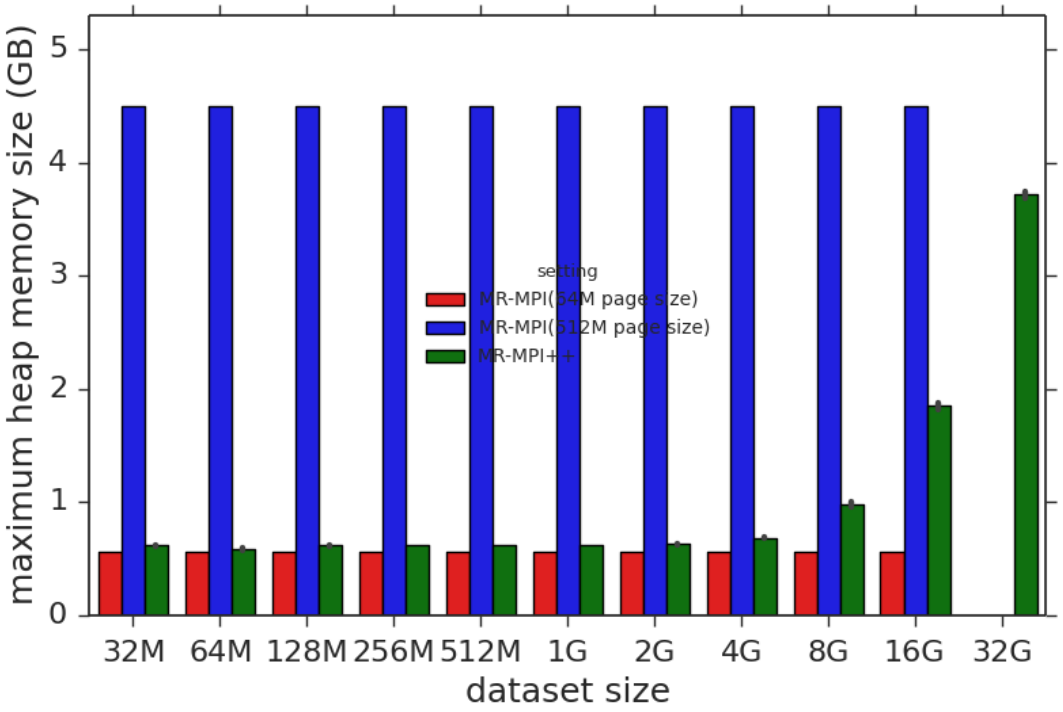




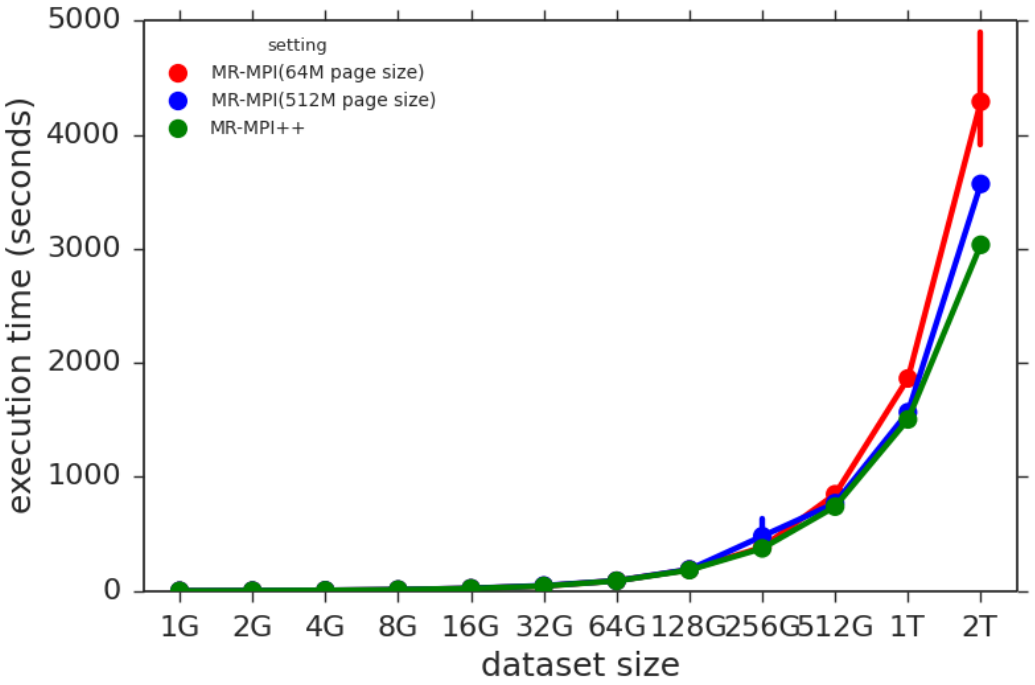
MR-MPI++ improves memory usage efficiency a lot.

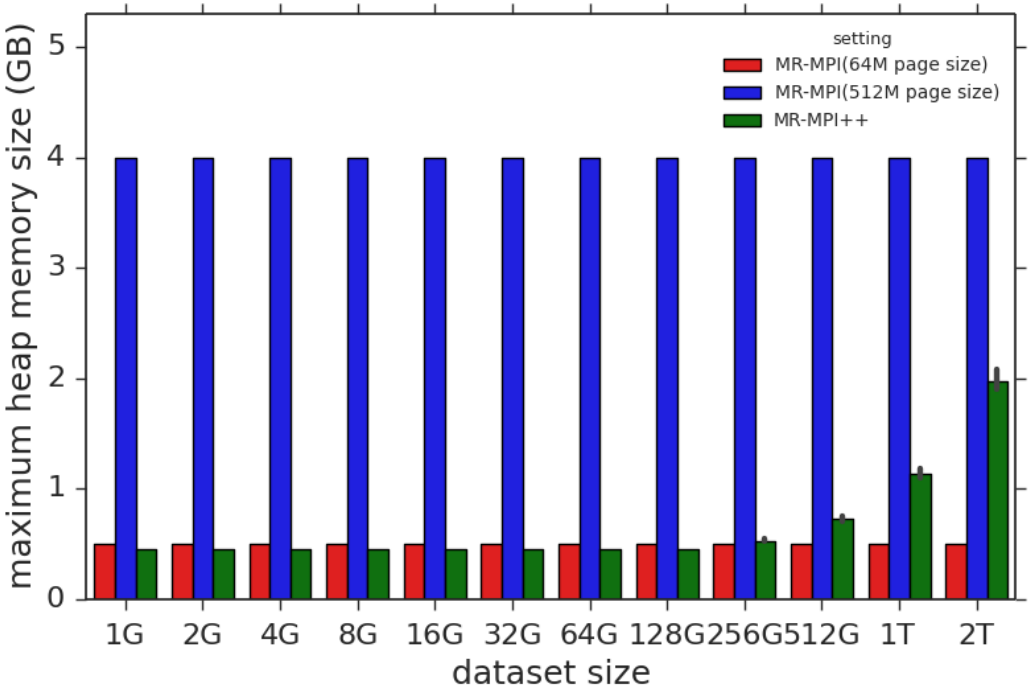
BFS


execution time (seconds)
1000
800
600
400
200
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
32M
64M
128M
256M
512M
1G
2G
4G
8G
16G
32G
dataset size


maximum heap memory size (GB)
5
4
3
2
1
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
32M
64M
128M
256M
512M
1G
2G
4G
8G
16G
32G
dataset size

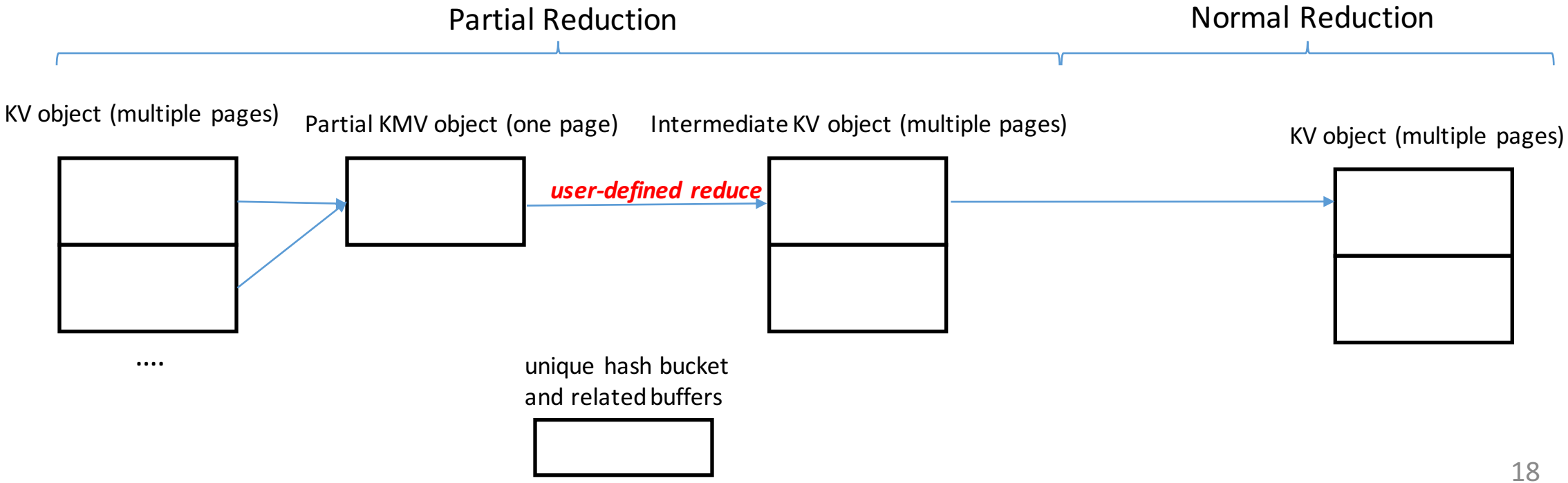
Octree


setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
execution time (seconds)
0
1000
2000
3000
4000
5000
1G
2G
4G
8G
16G
32G
64G
128G
256G
512G
1T
2T
dataset size


setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
maximum heap memory size (GB)
0
1
2
3
4
5
1G
2G
4G
8G
16G
32G
64G
128G
256G
512G
1T
2T
dataset size

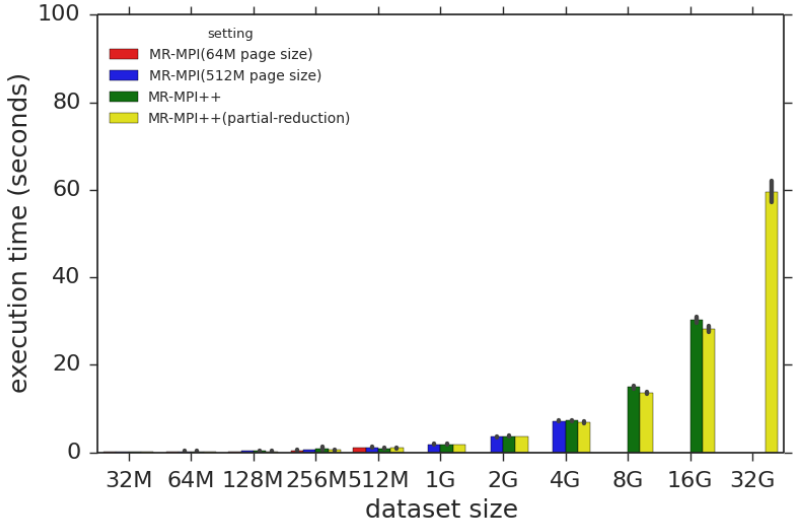
Partial-reduction: reduce memory requirement in reduce phase for some applications

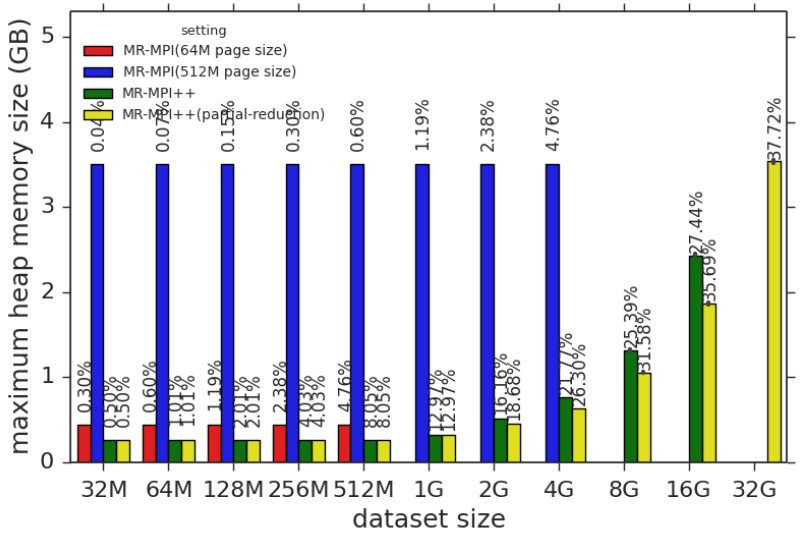
- Basic idea
  - For some applications (for example wordcount), we can apply reduce to partial KVs.
  - The reduced result of KVs is often much smaller than the original KVs.
  - As a result, we can apply partial reduction for each page and then perform the normal reduction with the intermediate results.



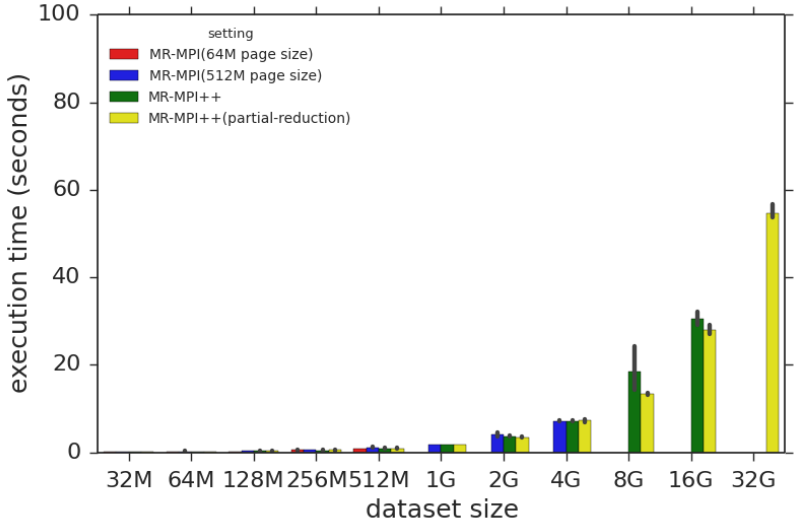
Wordcount Benchmark

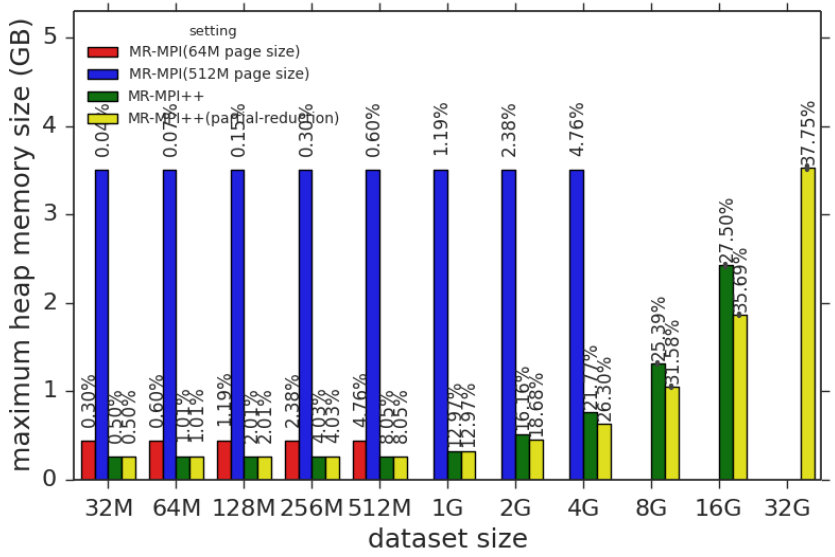
Uniform



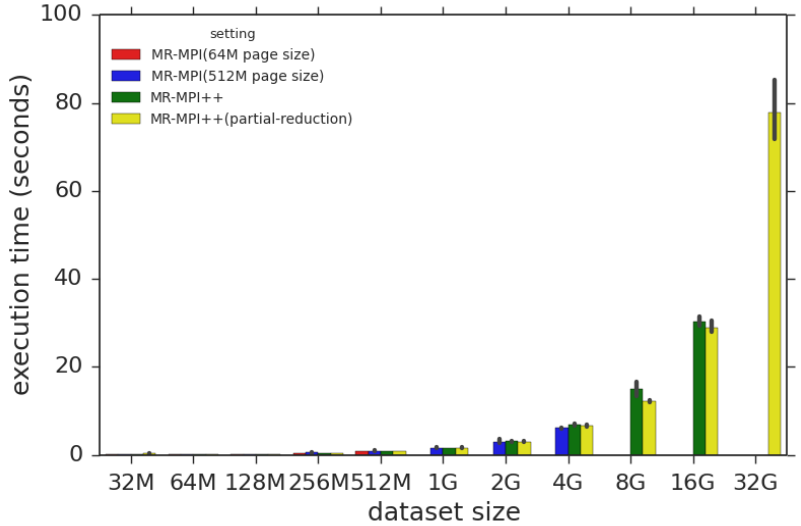


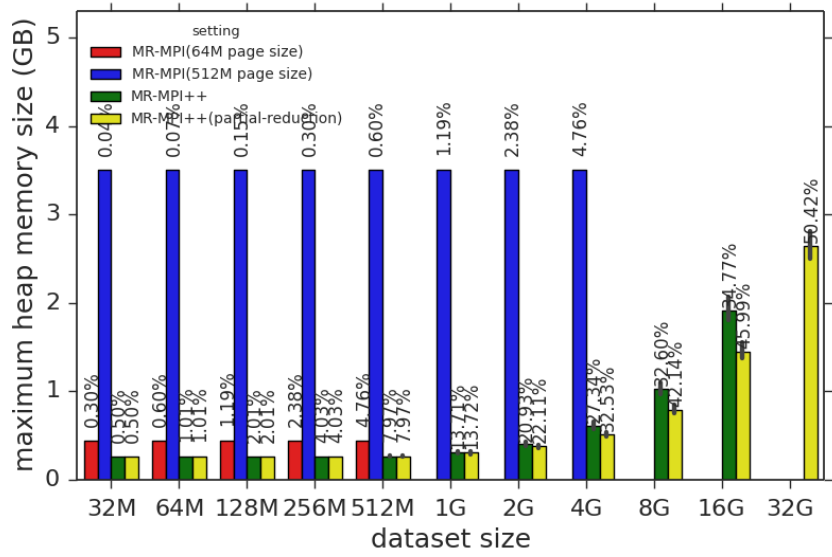
Nonuniform





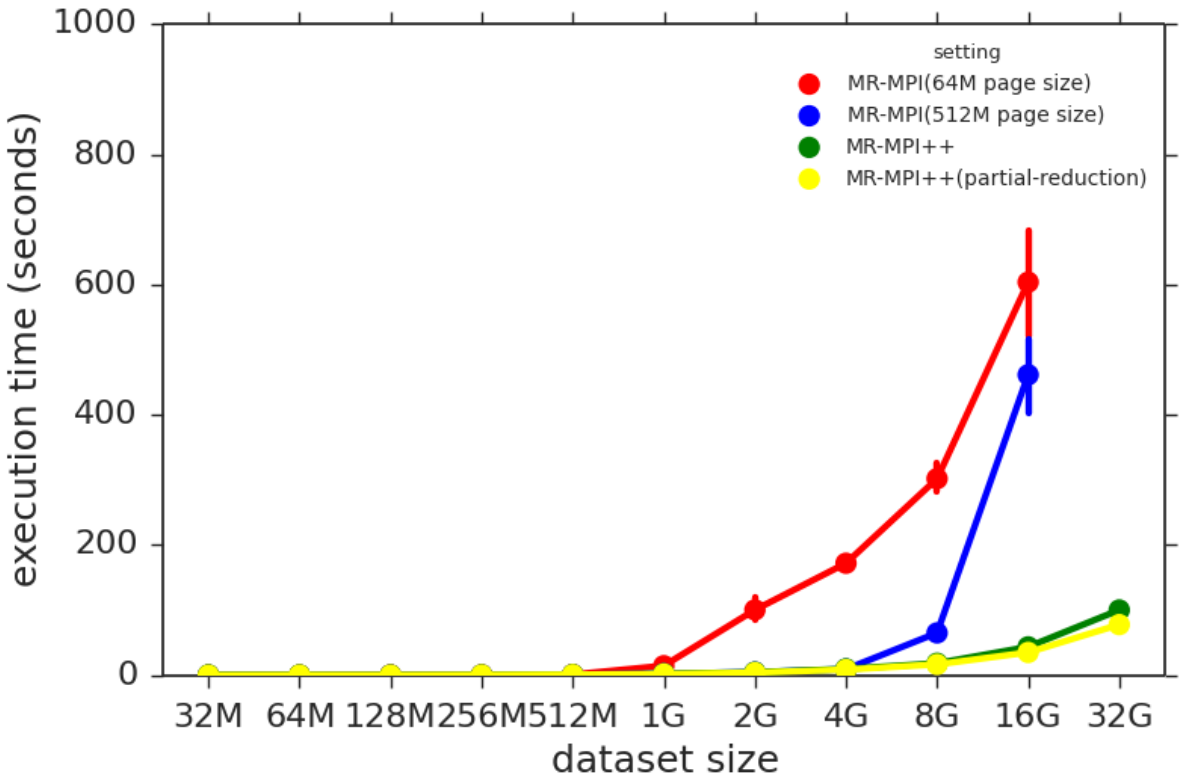
Wikipedia

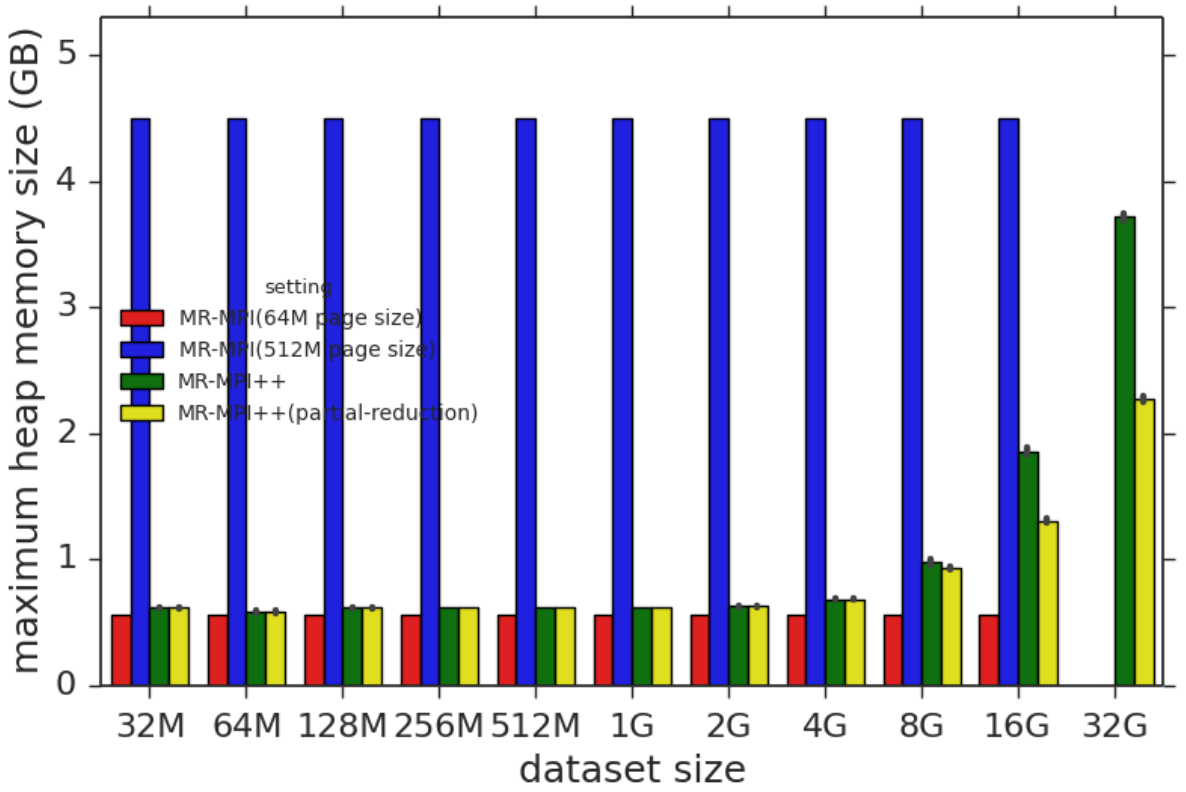




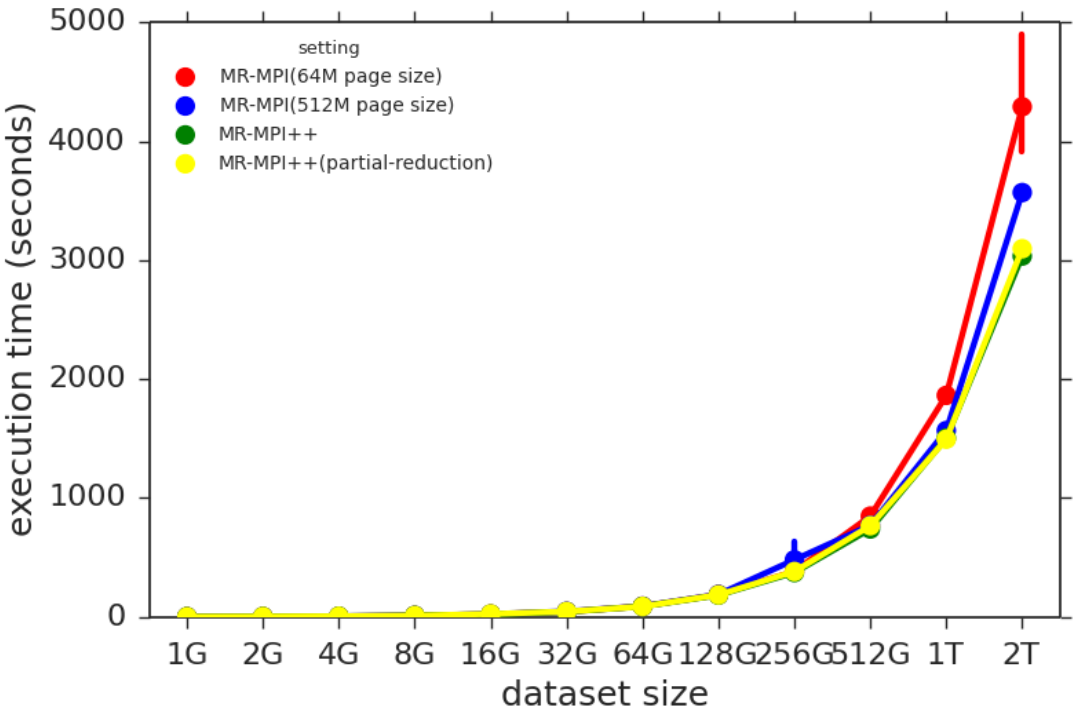
Partial-reduction improves memory usage efficiency further.

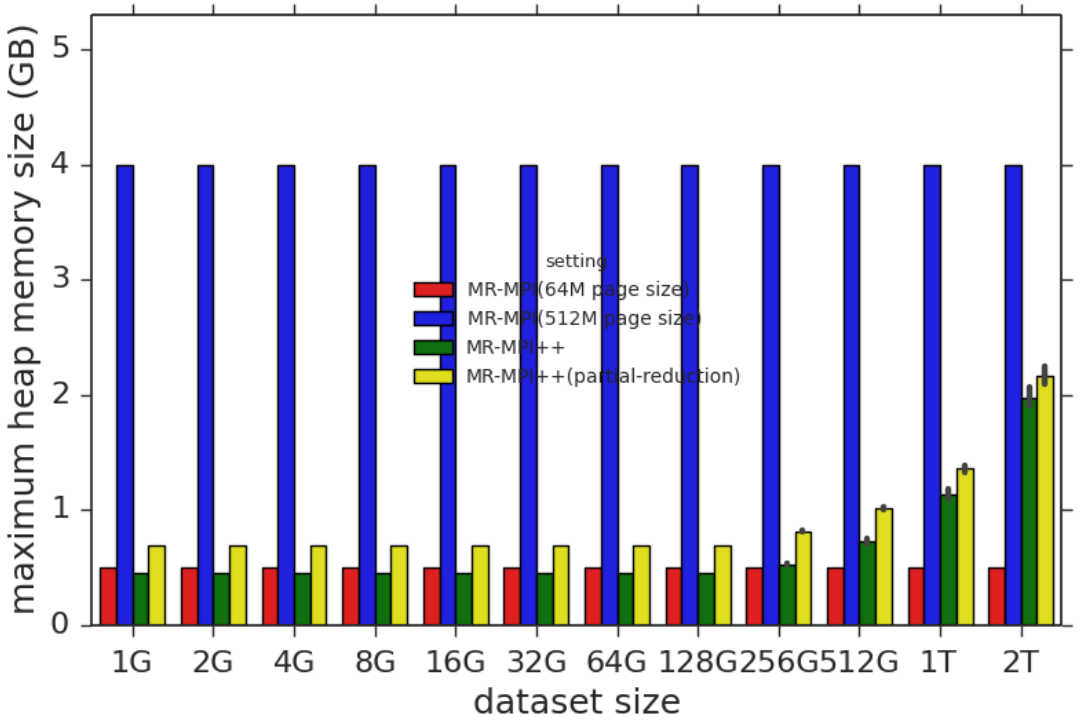
BFS


execution time (seconds)
1000
800
600
400
200
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
MR-MPI++(partial-reduction)
32M
64M
128M
256M
512M
1G
2G
4G
8G
16G
32G
dataset size


maximum heap memory size (GB)
5
4
3
2
1
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
MR-MPI++(partial-reduction)
32M
64M
128M
256M
512M
1G
2G
4G
8G
16G
32G
dataset size

Octree


execution time (seconds)
5000
4000
3000
2000
1000
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
MR-MPI++(partial-reduction)
1G
2G
4G
8G
16G
32G
64G
128G
256G
512G
1T
2T
dataset size


maximum heap memory size (GB)
5
4
3
2
1
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
MR-MPI++(partial-reduction)
1G
2G
4G
8G
16G
32G
64G
128G
256G
512G
1T
2T
dataset size

KV Representation Hint

- General KV

- String KV

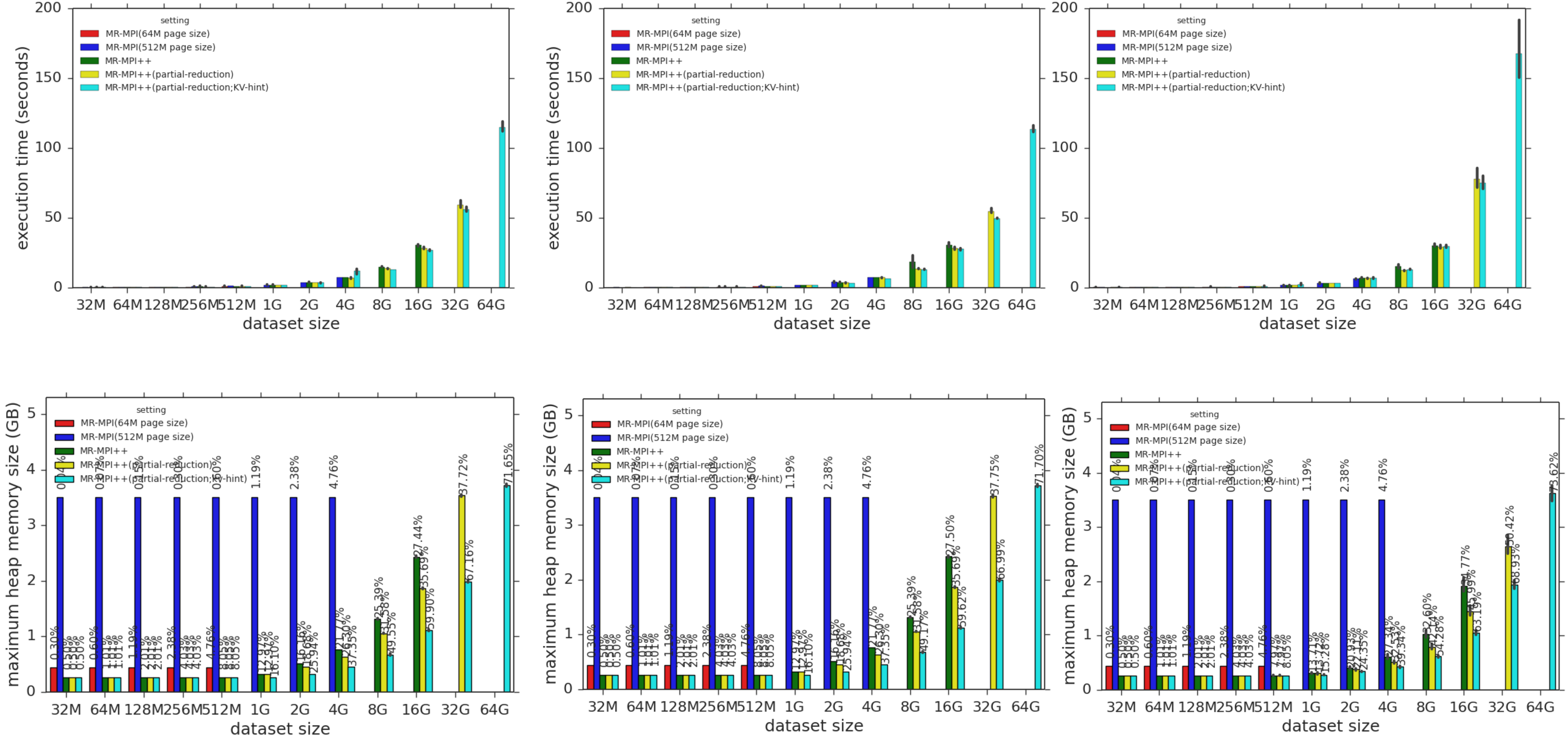
strlen() is used to get key and value size

- Fixed size KV

Fixed size of key and value

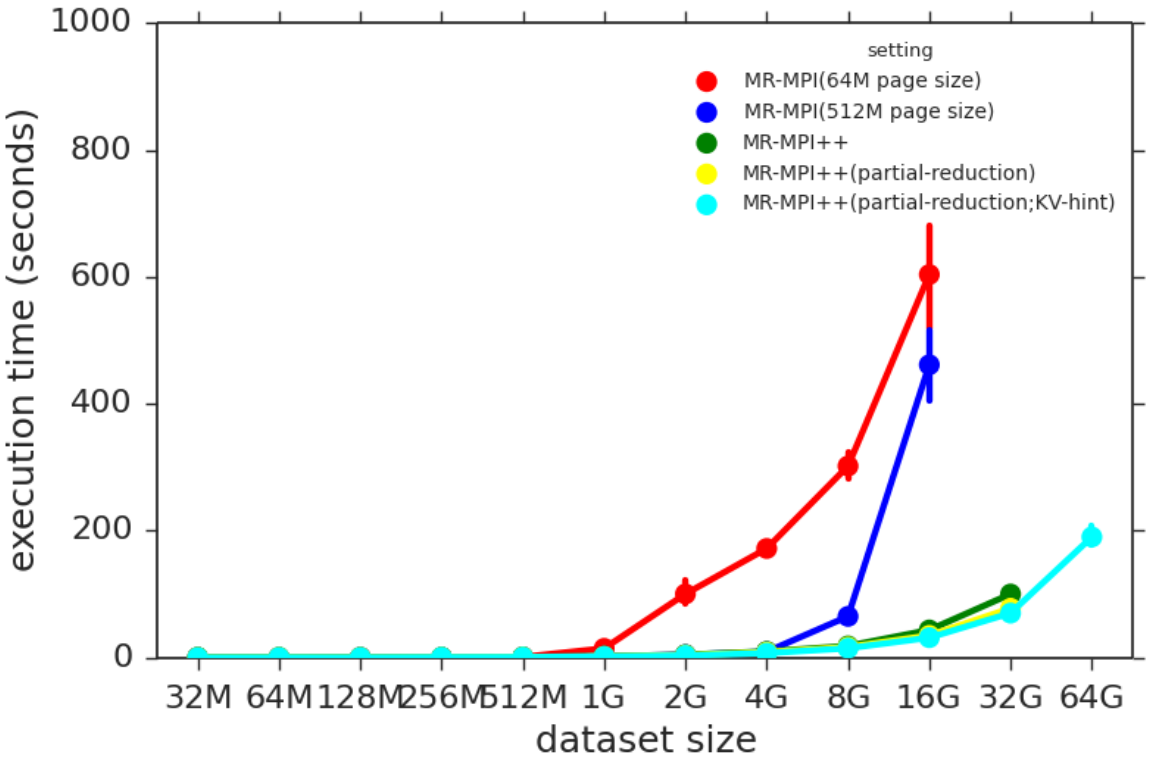
Wordcount Benchmark

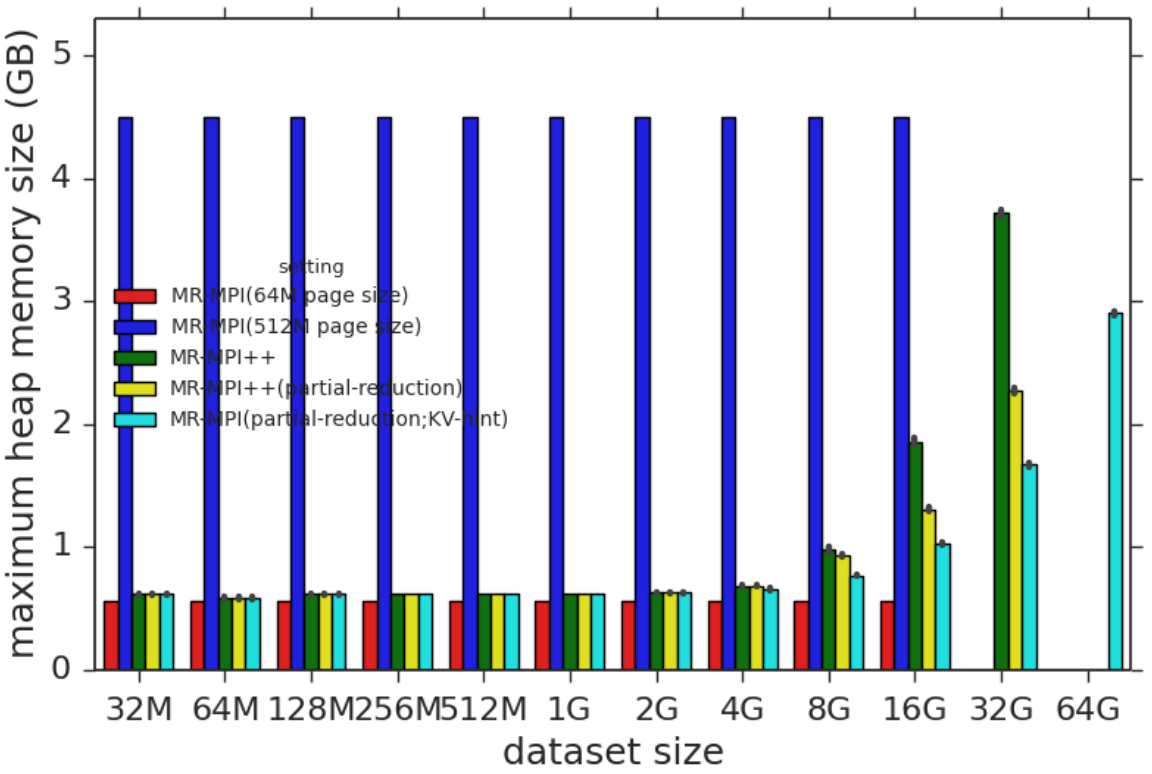
Uniform | Nonuniform | Wikipedia



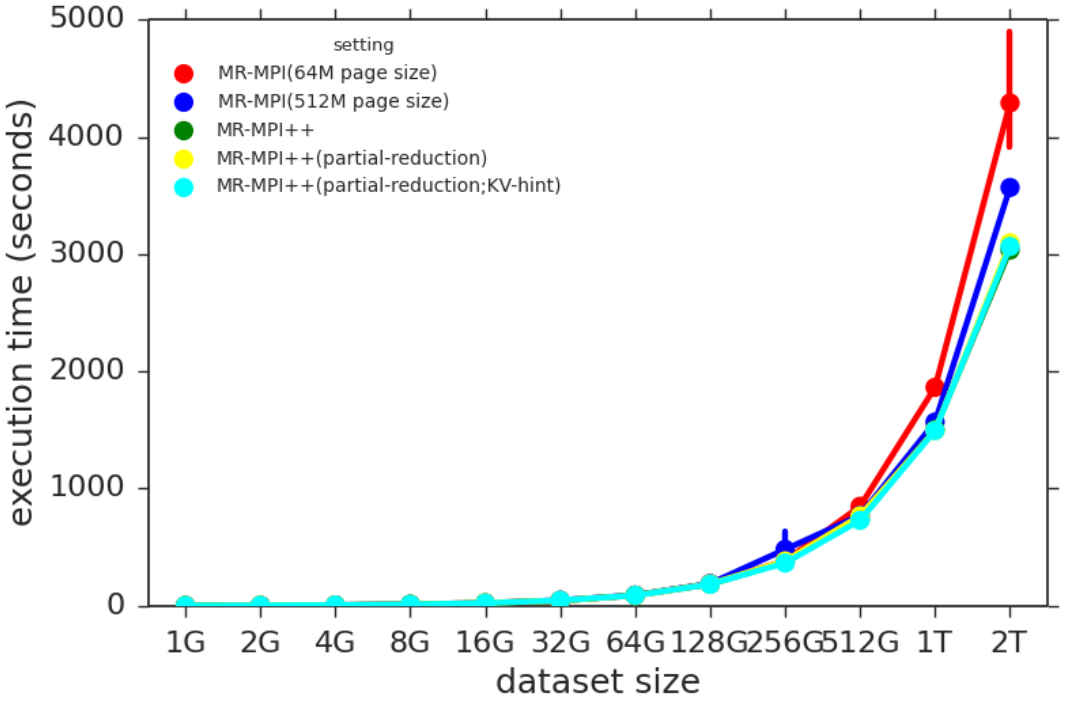
KV-hint optimization improves memory usage efficiency further.

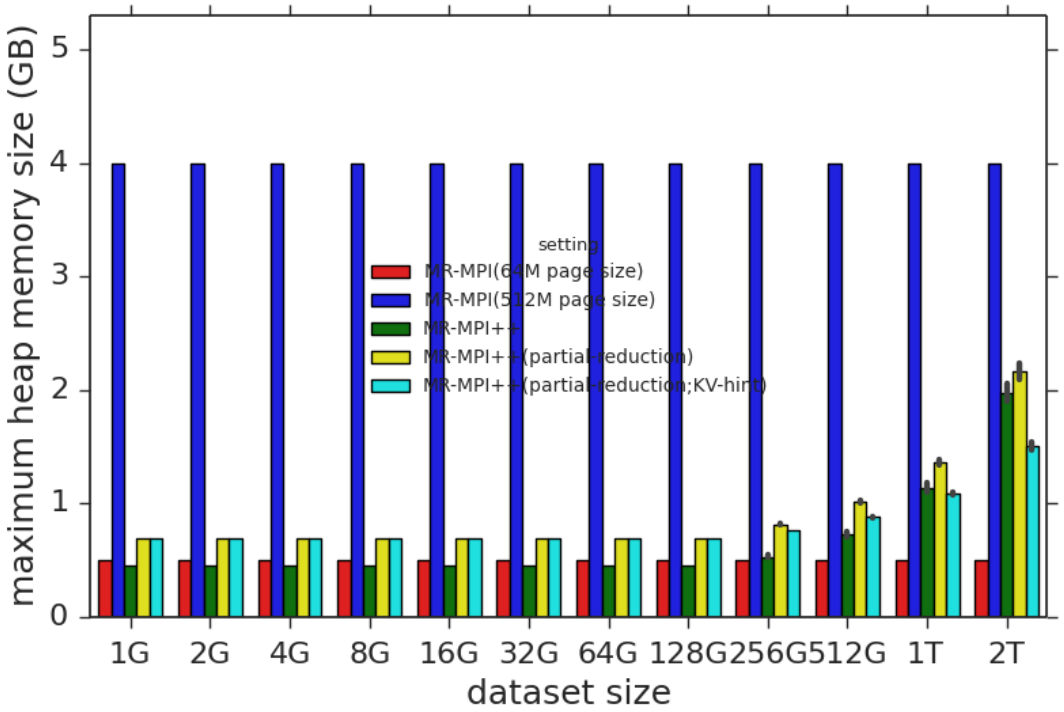
BFS


execution time (seconds)
1000
800
600
400
200
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
MR-MPI++(partial-reduction)
MR-MPI++(partial-reduction;KV-hint)
32M 64M 128M 256M 512M 1G 2G 4G 8G 16G 32G 64G
dataset size

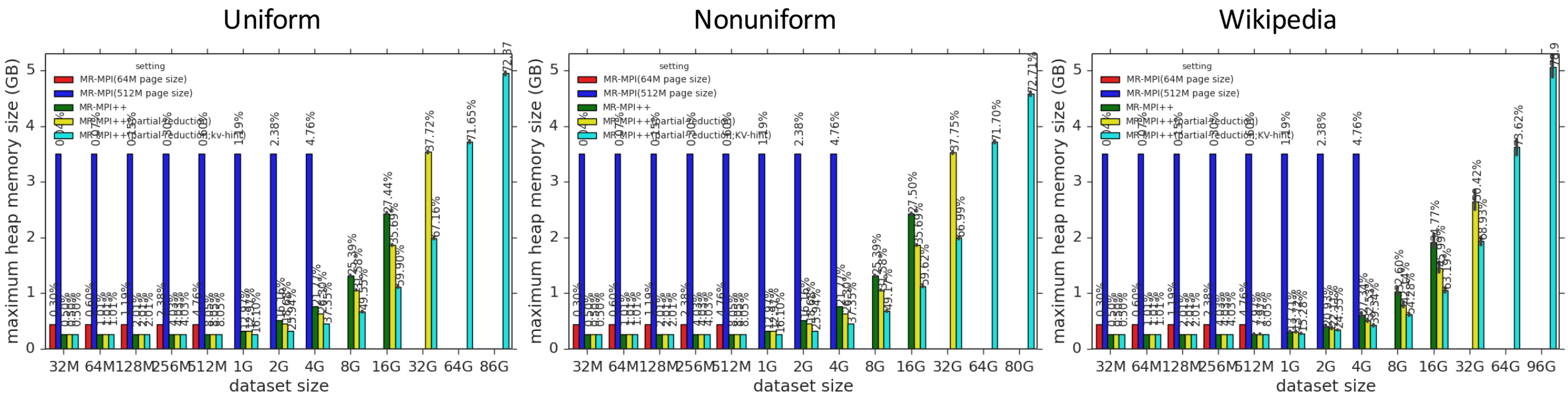

maximum heap memory size (GB)
5
4
3
2
1
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
MR-MPI++(partial-reduction)
MR-MPI(partial-reduction;KV-hint)
32M 64M 128M 256M 512M 1G 2G 4G 8G 16G 32G 64G
dataset size

Octree


execution time (seconds)
5000
4000
3000
2000
1000
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
MR-MPI++(partial-reduction)
MR-MPI++(partial-reduction;KV-hint)
1G
2G
4G
8G
16G
32G
64G
128G
256G
512G
1T
2T
dataset size


maximum heap memory size (GB)
5
4
3
2
1
0
setting
MR-MPI(64M page size)
MR-MPI(512M page size)
MR-MPI++
MR-MPI++(partial-reduction)
MR-MPI++(partial-reduction;KV-hint)
1G
2G
4G
8G
16G
32G
64G
128G
256G
512G
1T
2T
dataset size

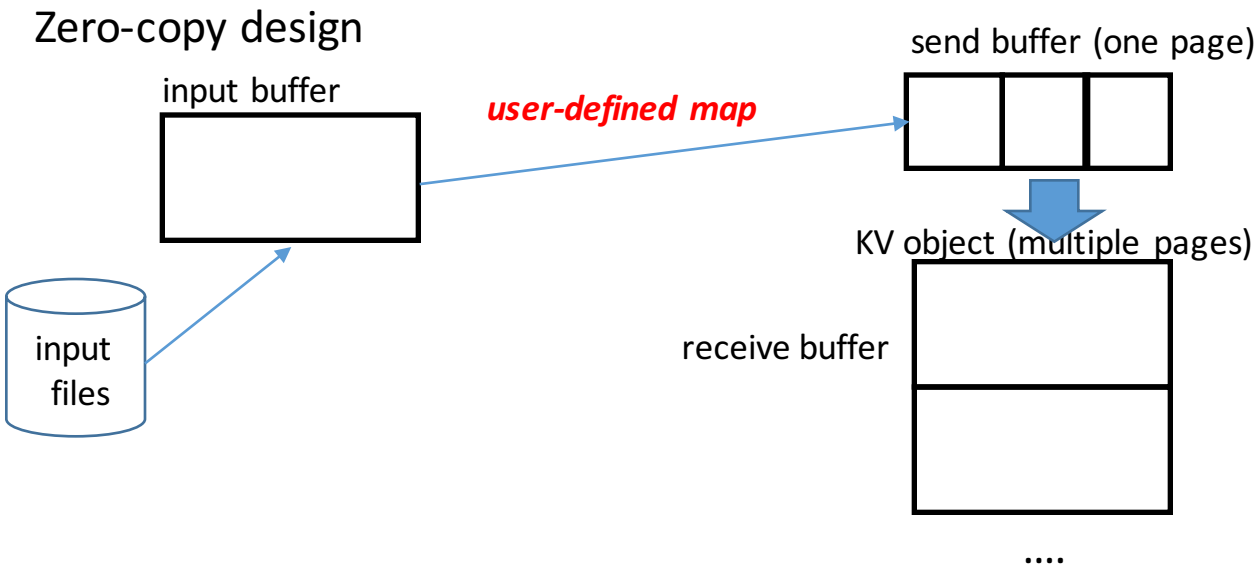
Maximum Memory Usage Efficiency



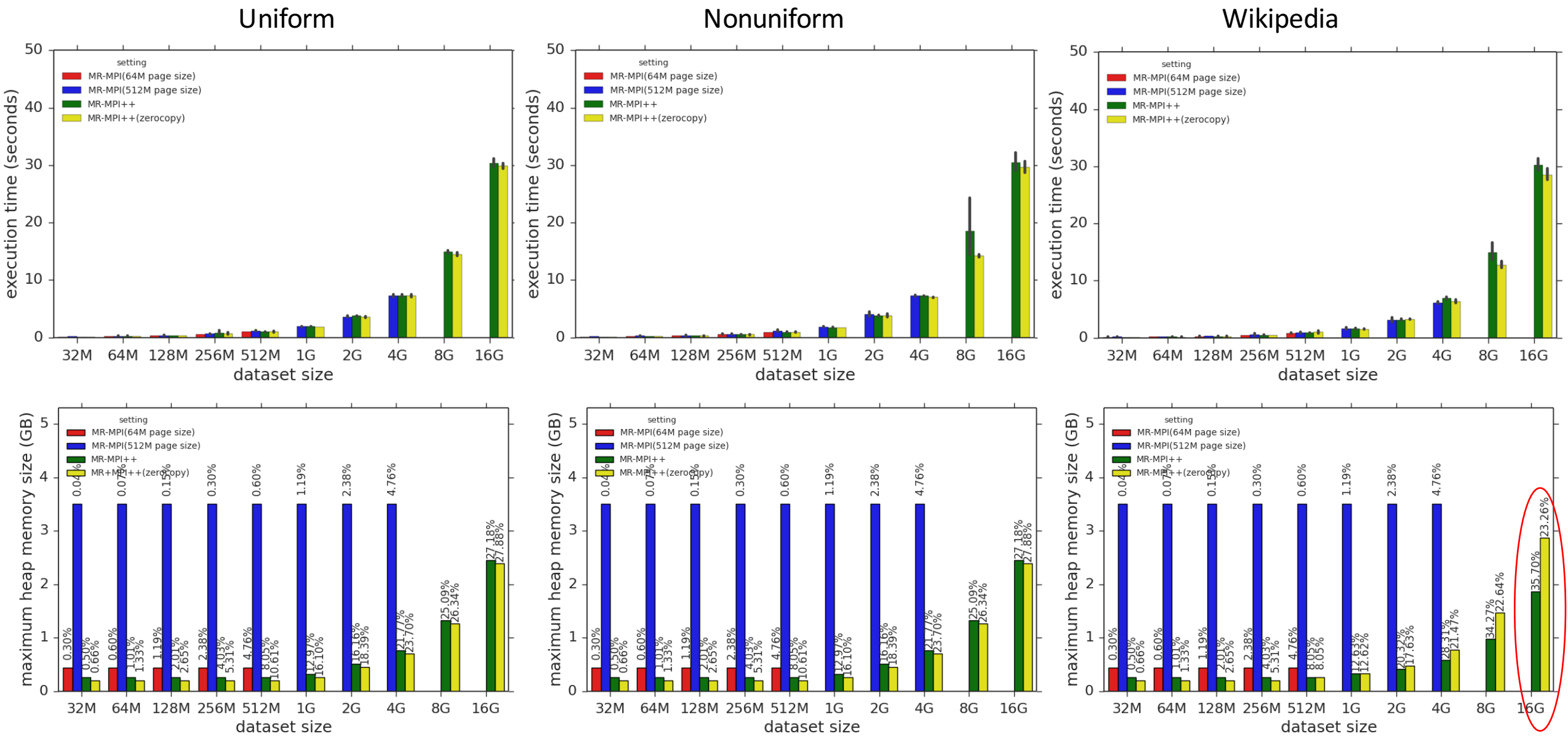
This data may be not the best, the slurm configuration limit the maximum memory size used to 120GB.

Problems of Zero-copy Design

- Basic Ideas
  - Use buffers of KV object as receive buffers directly (reduce memory size and memory copy)



Issue Description:
1. The pages in the KV object use the same buffer size.
2. Each time, one page is allocated to receive the results of MPI_Alltoallv.
3. If the received data size is not balanced, the process whose received data is much less than the page size will waste a lot memory.



1. Zero-copy design doesn't performance very much.
2. For the imbalanced (wikipedia) dataset, zero-copy implementation wastes memory.

In the future, ignore the zero-copy optimization?