HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Transactional In-memory Storage for Extended MapReduce

Kim-Thomas Rehmann and Michael Schöttner
Heinrich-Heine Universität Düsseldorf
Abteilung Betriebssysteme

Outline

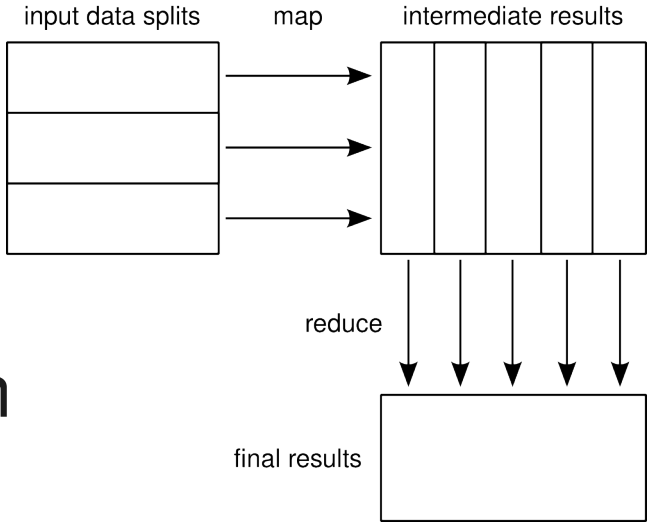
HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Extended MapReduce
- In-memory MapReduce
- The ECRAM In-memory Storage
- EMR Framework and Applications
- Evaluation
- Conclusion

MapReduce

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Programming model for parallel computations
  - *Embarrassingly parallel* map and reduce phase
  - Simplifies parallelization, failure handling, synchronization

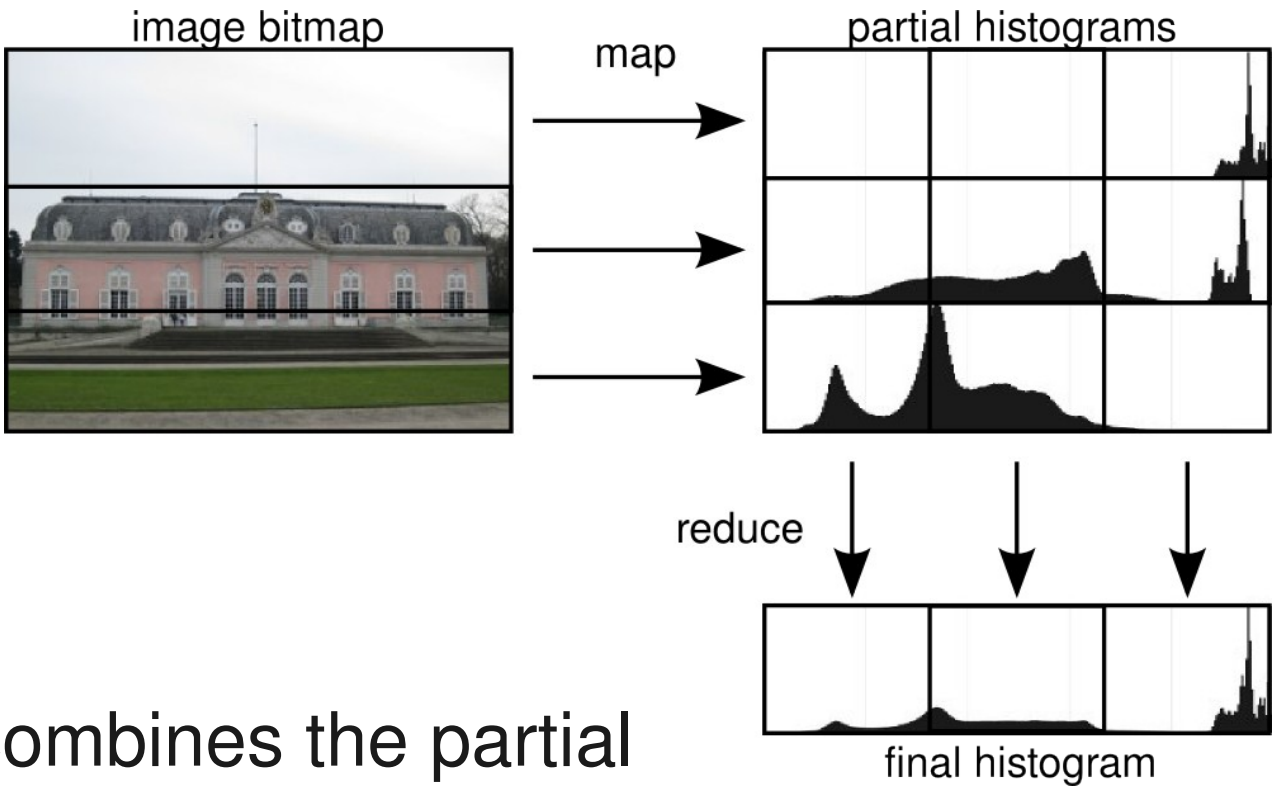


- Data access characteristics
  - Constant input data
  - Intermediate, final results are append-only
  - Benefits from GFS's append-at-least-once operation

MapReduce: Histogram

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Map phase
  - Each *map* job computes a partial histogram for an image split
- Reduce phase
  - Each *reduce* job combines the partial histograms for a color interval



Extended MapReduce

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Extends the *single-pass* model
  - Iterative computations, on-line stream processing
  - Updates input data, intermediate and/or final results
  - Complicates consistency handling
- Existing frameworks
  - Use message-passing instead of a distributed filesystem
    - e.g. multicasting and publish-subscribe (Twister)
    - e.g. pipelining data from producer to consumer (HOP)
  - Deviate from original data-centric model

In-memory MapReduce

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Data-oriented communication model for extended MapReduce
  - Not need to send messages
  - Keep all data in RAM
  - Guarantee consistency (e.g. using DTM)
  - Allows for adaptive replication and fault-tolerance
- Prototype framework: extended in-memory MapReduce (EMR) based on ECRAM

ECRAM In-memory Storage

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Distributed objects for Cloud computing
  - All data in RAM for fast accesses
  - Multi-version objects with adaptive consistency and replication
- Object access
  - Accessor/mutator functions
  - Memory-mapping
  - Key-based routing of requests to manager nodes (similar to a DHT)

ECRAM Transactions

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Optimistic synchronization of transactions
  - Speculative execution, transparent restart
  - Flat nesting allowed
- Validation
  - Backward validation: central validator node checks serializability and assigns transaction IDs
  - Forward validation: Each node receives commit notifications to detect conflicts early

ECRAM Weak Consistency

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Accesses outside transactions
  - Useful if the programmer can preclude conflicts
  - Read accesses may return out-of-date content
- Opaque validation
  - Non-validated accesses
  - Voluntary restart, limit maximum number of restarts
- Local commits avoid validation
  - For read-only transactions
  - For transactions that access non-replicated objects only

ECRAM Replication

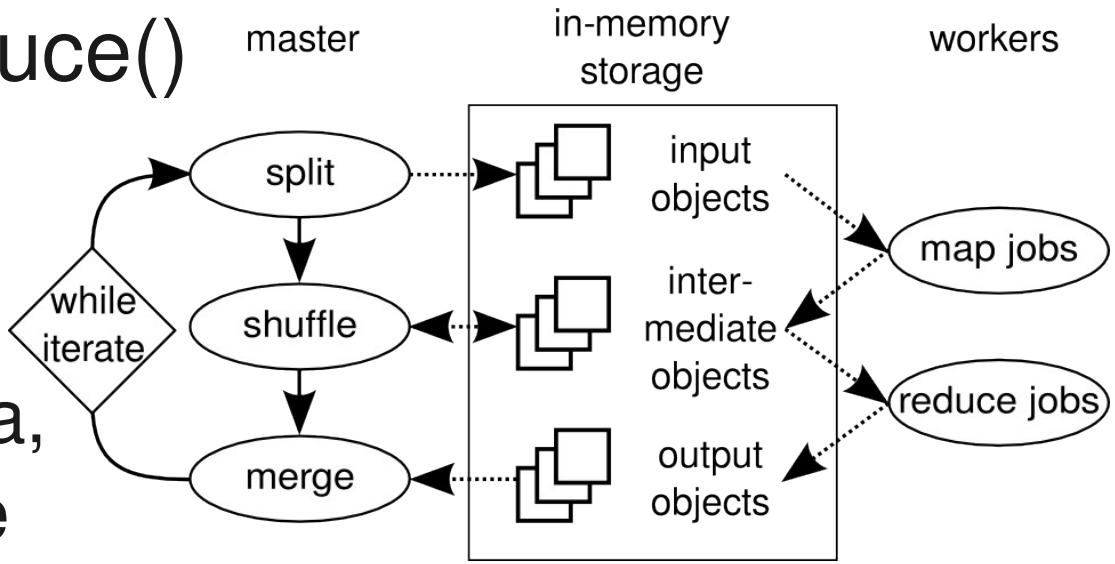
HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Orthogonal to consistency
- Multi-version support
  - Always guarantees a consistent view
  - Allows weakly consistent accesses
  - Validator determines which old versions to discard
- Adaptive coherency
  - Switches between invalidate and update protocol
  - Based on object access monitoring

EMR Execution Model

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Shared data stored in ECRAM
  - Input, intermediate and output objects
  - Internal data such as jobs and nodes
- Parameters to mapreduce()
  - map and reduce functions (required)
  - How to parse input data, split, shuffle and merge functions, termination condition for iteration etc. (optional)



EMR Job Scheduler

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Shared job lists and worker queues
  - Handle execution dependencies as data dependencies
  - Synchronize master and workers using *"condition variables"*
  - e.g. submitting jobs and waiting for their completion

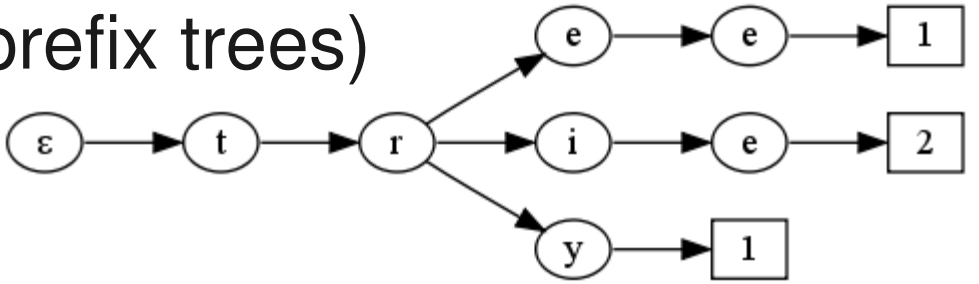
```
for (job = 0; job < nmaps; job ++)
{
    ecram_bot();
    job_submit(queue, input_split, app->map, data,
        &app->ncompleted_maps, 1);
    ecram_eot();
}
ecram_wait(&app->ncompleted_maps, 0, wait_equal, nmaps, timeout);
```

- Load balancing based on work-stealing

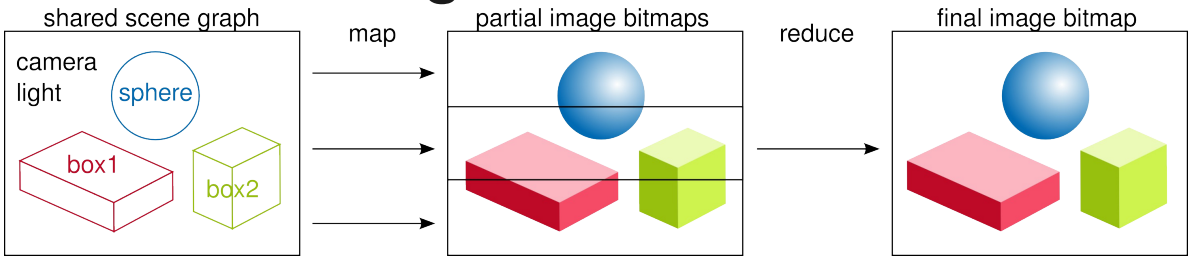
Applications

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Classical MapReduce example: word-counting
  - Implemented using tries (prefix trees) to reduce collisions



- Real-time raytracing
  - Map: each worker calculates an image region
  - Reduce: one worker combines regions and saves the result



- K-Means clustering
  - Iterative approximation of n cluster centers

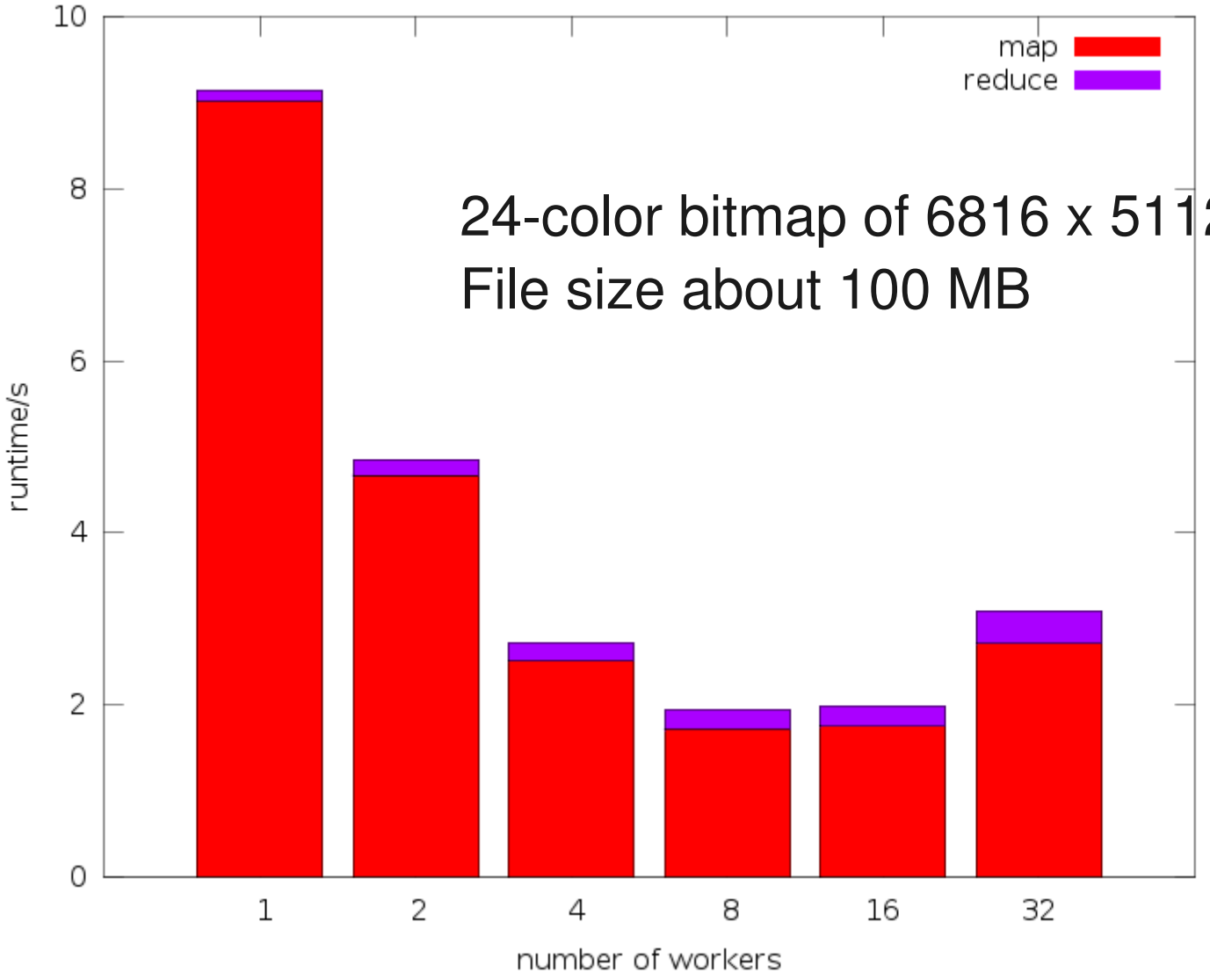
Evaluation – Testbed

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- 33 AMD Opteron dual-CPU nodes
  - Opteron (17 x 246 @ 2 GHz, 16 x 244 @ 1.8 GHz)
  - Each 2 GB ccNUMA RAM
  - Debian 6.0 Squeeze with Linux 2.6.32 kernel
  - Disk-less clients, booted via NFS, GB ethernet
- Experiments
  - Latency of map and reduce phases
  - Framerate of iterative MapReduce
  - Latency and bandwidth for adaptive replication

Evaluation – Latency of Histogram Computation

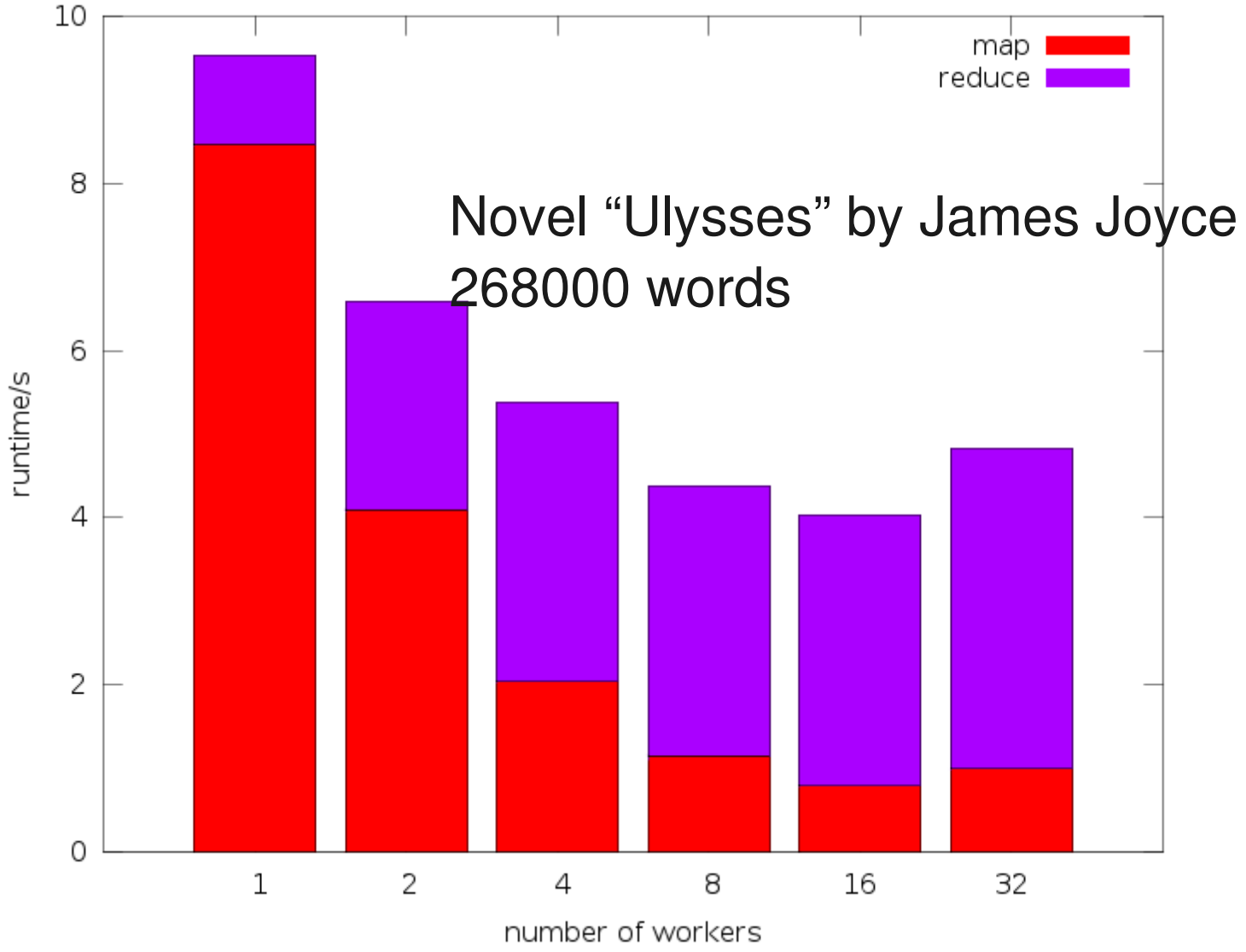
HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF



24-color bitmap of 6816 x 5112 pixel
File size about 100 MB

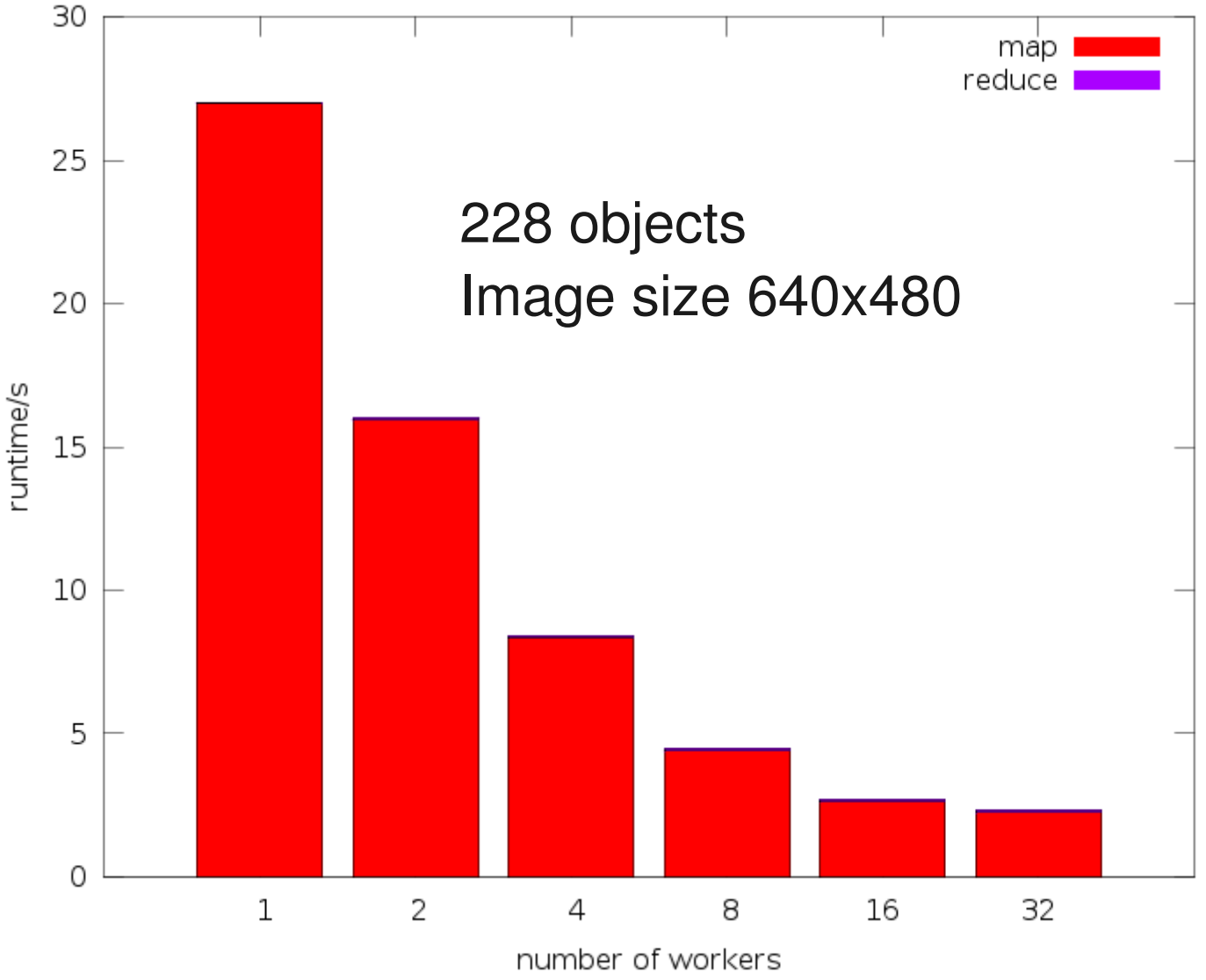
Evaluation – Latency of Wordcount

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF



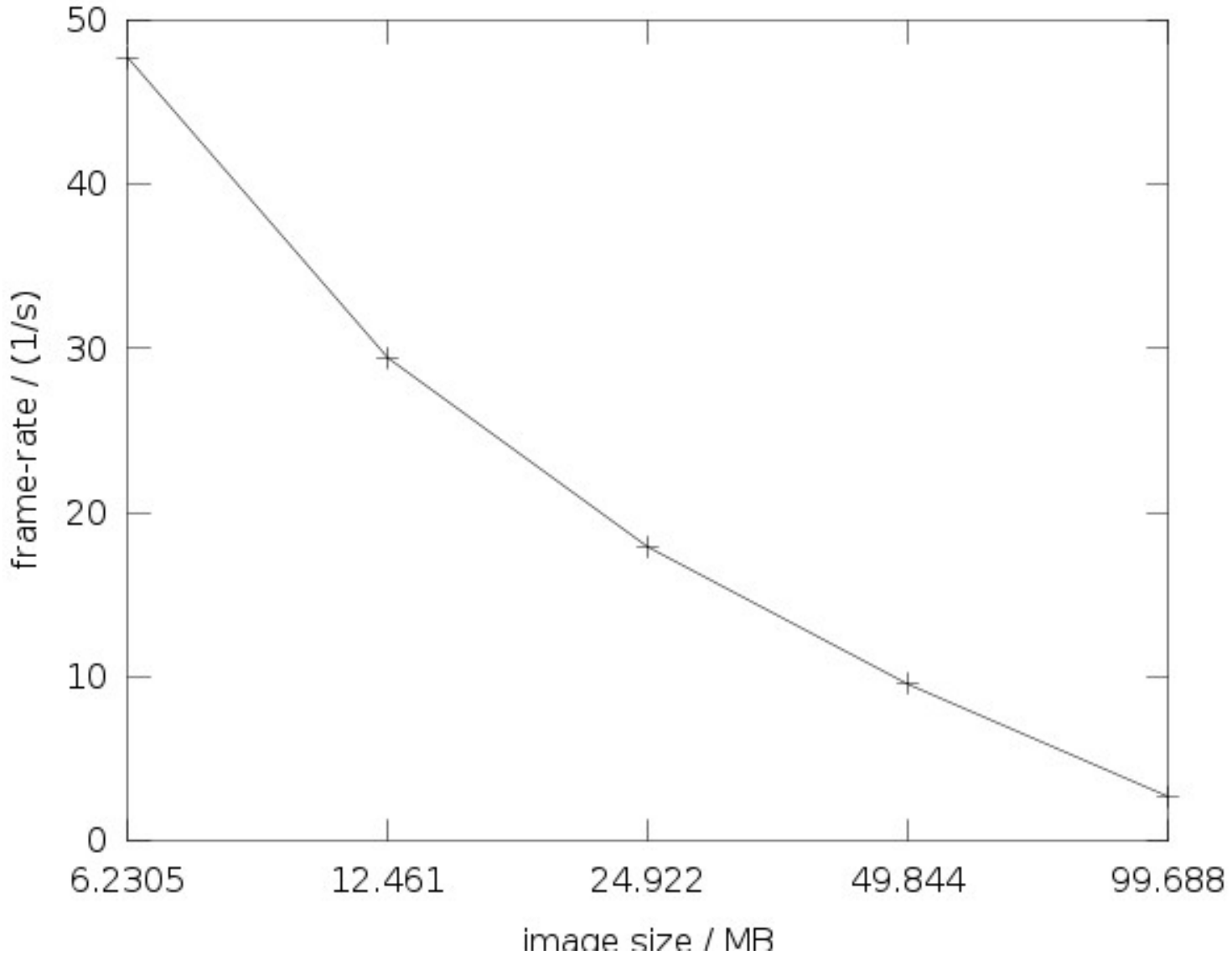
Evaluation – Latency of Raytracing

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF



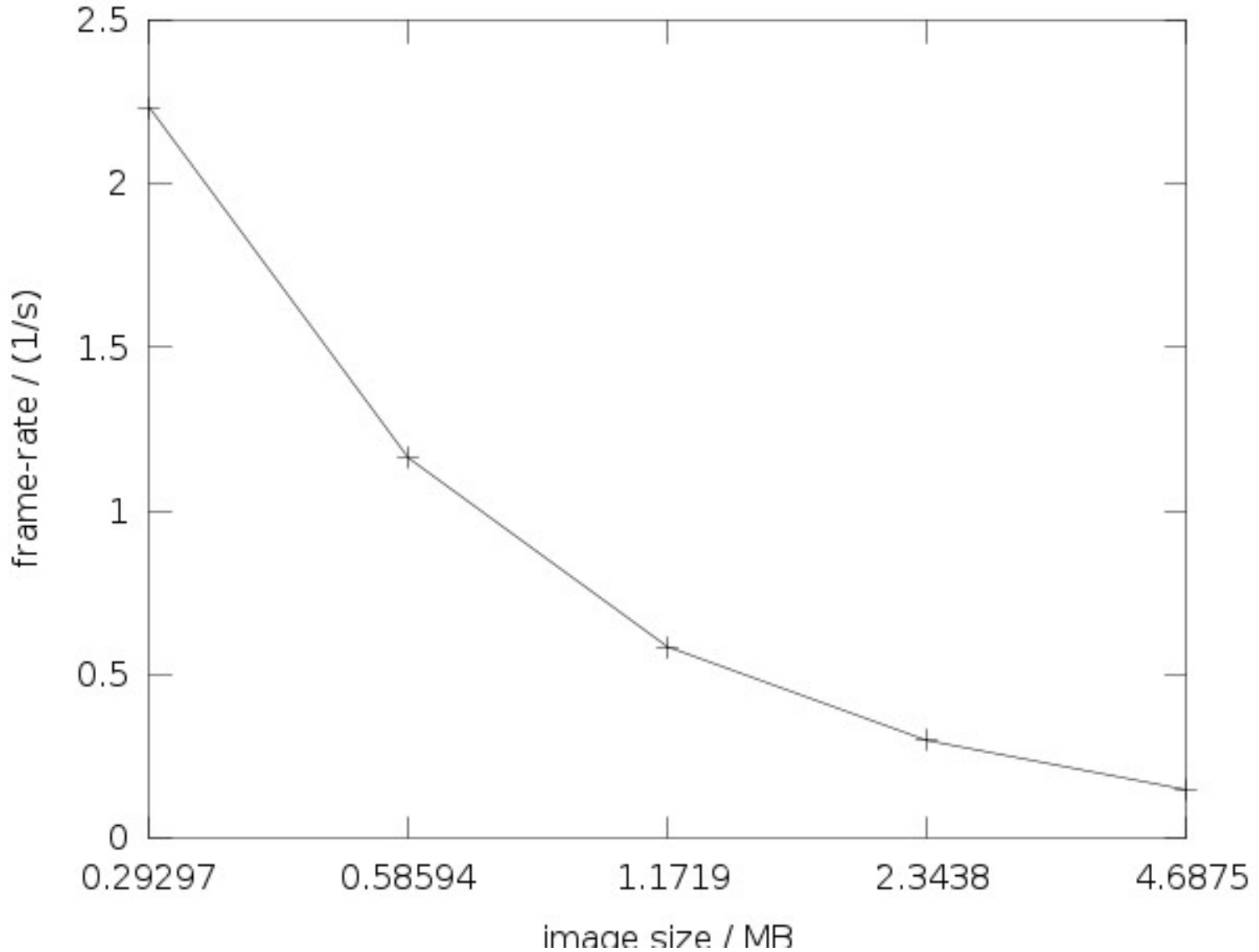
Evaluation – Framerate of Iterative Histogram

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF



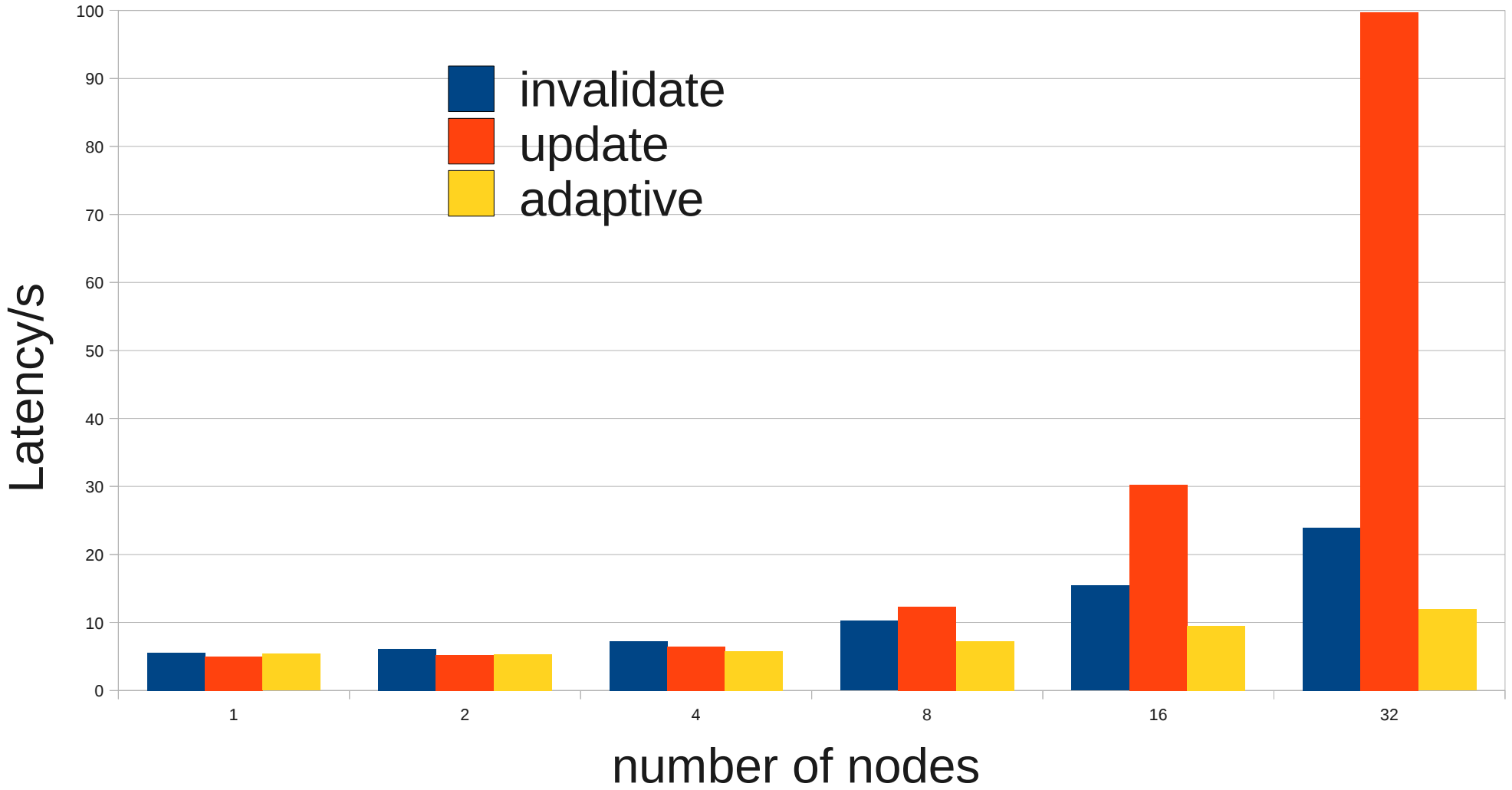
Evaluation – Framerate of Iterative Raytracer

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF



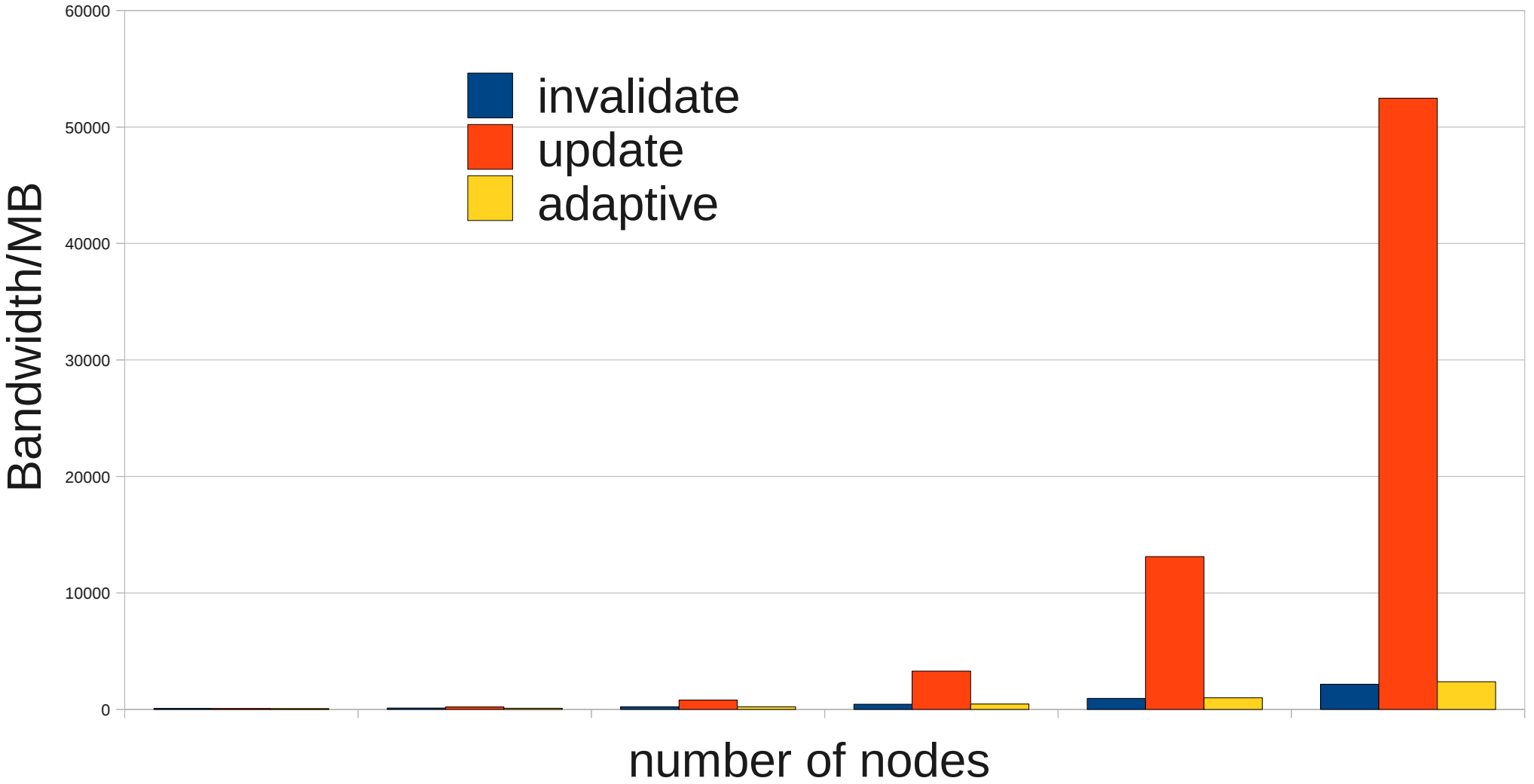
Evaluation – Latency of Adaptive Replication

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF



Evaluation – Bandwidth Requirement of Replication

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF



Conclusion

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- In-memory MapReduce model:
  Data-oriented communication for extended MapReduce
- In-memory job scheduler supporting load balancing
- Development of EMR applications benefits from ECRAM storage
  - In-memory objects
  - Transactions, weak consistency, adaptive replication