Benchmarks

- Hama 0.7.0-SNAPSHOT vs Giraph 1.2.0
- Graph Package: PageRank Benchmarks 0.6.4 vs 0.7.0-SNAPSHOT
- PageRank Benchmark test 0.6.4 vs 0.6.3
- Hama 0.6 RC3
- Compressor Comparison Table
- PageRank
- Compare with old version
- Compare between Hadoop RPC and Avro
- PageRank (PR)
- Compare with old version
- K-Means Clustering
- Single Shortest Path Problem
- Random Communication Benchmark
- 4 racks
  - Hama 0.4
- 1 rack
  - Hama 0.4 (r.1177507)
  - Hama 0.3
  - Hama 0.2
    - Test 1 (many small messages vs. few large messages)
    - Test 2

Hama 0.7.0-SNAPSHOT vs Giraph 1.2.0

- 2 node cluster
  - Quanta S210 X22RQ-3
  - CPU: Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz x 2
  - MEM: 192 GB
  - HDD: 3TB x 1 (HW RAID, Physical HDD x 2)
  - NIC: 1G x 2, 10G x 4
- Same input data (Random generated JSON format text files) was used
- Same child opts (-Xmx4048m)
- Same number of tasks was used
- Same hash partitioner was used

NOTE: Hama allows you to generate a random graph data in JSON format, which is easily compatible with other similar systems. Try it on your cluster!

Vertices	The max-edges per vertex	Tasks	Hama-0.7.0	Giraph-1.2.0
300000	1000	30	70.425 seconds	154 seconds
300000	900	30	64.388 seconds	141 seconds
300000	800	30	58.355 seconds	130 seconds
300000	700	30	49.309 seconds	114 seconds
300000	600	30	44.334 seconds	104 seconds
300000	500	30	40.321 seconds	92 seconds
300000	400	30	37.304 seconds	81 seconds
300000	300	30	31.248 seconds	72 seconds

Vertices	The max-edges per vertex	Tasks	Hama-0.7.0	Giraph-1.2.0
30000	10000	20	72.485 seconds	130 seconds
30000	9000	20	70.399 seconds	124 seconds
30000	8000	20	58.388 seconds	113 seconds
30000	7000	20	49.39 seconds	104 seconds
30000	6000	20	47.153 seconds	89 seconds
30000	5000	20	40.34 seconds	79 seconds

30000	4000	20	37.292 seconds	72 seconds
30000	3000	20	31.483 seconds	61 seconds

What are the major changes from the last release?

The major improvement changes are in the queue and messaging systems. We now use own outgoing/incoming message manager instead of using Java's built-in queues. It stores messages in serialized form in a set of bundles (or a single bundle) to reduce the memory usage and RPC overhead. Another important improvement is the enhanced graph package. Instead of sending each message individually, we package the messages per vertex and send a packaged message to their assigned destination nodes. The thread-pool executor service also used for each vertex computation. With this, we achieve better performance.

Graph Package: PageRank Benchmarks 0.6.4 vs 0.7.0-SNAPSHOT

- In 0.7.0-SNAPSHOT, the sender-side sends a list of messages per Vertex, the receiver-side uses normal Queue instead of Sorted Queue
  - CPU and Memory usage has been dramatically decreased!
- Kryo Serializer used for message serialization

https://lh3.googleusercontent.com/-AjfoPL8c-I0/VRI4metjElI/AAAAAAAAE84/1IkhjzweP1Q/w778-h470-no/hama_1.png

	Input Size (Bytes)	Total Tasks	Total Edges	Execution Time (0.6.4)	Execution Time (0.7.0-SNAPSHOT)
pagerank	905310	6	200,000	16.71 seconds	13.598 secs
pagerank	7065912	6	2,000,000	19.514 seconds	16.524 seconds
pagerank	857467170	6	200,000,000	1133.622 seconds	202.996 seconds
pagerank	857467170	10	200,000,000	770.589 seconds	145.751 seconds
pagerank	1437500140	10	300,000,000	1337.851 seconds	229.847 seconds

PageRank Benchmark test 0.6.4 vs 0.6.3

See http://blog.datasayer.com/2014/03/apache-hama-benchmark-test-at-lg-cns.html

Hama 0.6 RC3

- Oracle BDA
- 180 Tasks
- SDP protocol disabled

	Conditions	Execution Time
bench	16 10000 32	21.278 seconds
bench	16 100000 32	72.318 seconds
sssp	1 billion edges	473.52 seconds
pagerank	Google web graph dataset	9.15 seconds

- HAMA-664

	Conditions	Execution Time
bench	16 10000 32	15.266 seconds
bench	16 100000 32	51.293 seconds

Compressor Comparison Table

- SSSP job on Random Graph
- 153 Tasks
- 2 Oracle BDAs cluster

Compressor	100 million edges	200 million edges	400 million edges	800 million edges
None	96.424 seconds	129.397 seconds	159.408 seconds	336.502 seconds
Bzip2Compressor	112.592 seconds	198.411 seconds	387.558 seconds	672.663 seconds

- The first major part of the problem is inefficient calculation of bundle size. (CompressionUtil.getCompressionRatio() method)

PageRank

- 1 Rack Oracle BDA
- The dataset contains 5,716,808 pages and 130,160,392 links and is unzipped ~1gb large.

Tasks	Job Execution Time
170	18.465 seconds

Compare with old version

- 1 Rack Oracle BDA was used.

Version	Physical Nodes	Tasks	MSG bytes	MSG exchanges per Superstep	Supersteps	Job Execution Time
Hama 0.5	18	162	16	1,600,000	32	93.322 seconds
Hama 0.4-incubating	18	162	16	1,600,000	32	288.435 seconds

Compare between Hadoop RPC and Avro

Messenger	Physical Nodes	Tasks	MSG bytes	MSG exchanges per Superstep	Supersteps	Job Execution Time
Hadoop RPC	16	100	16	10 million	32	168.571 seconds
Avro RPC	16	100	16	10 million	32	159.573 seconds
Hadoop RPC	16	100	16	10,000	2048	711.924 seconds
Avro RPC	16	100	16	10,000	2048	735.994 seconds

PageRank (PR)

- Experimental environments
  - One rack (9 nodes 144 cores) cluster
  - 10G network
  - Hadoop 0.20.2
  - Hama 0.4 TRUNK r1235586.
  - MR based Random Graph Generator for Pagerank
  - Task and data partition based on hashing of vertextID in graph and size of input data.

Web pages (10 anchors per page)	Tasks	Supersteps	Job Execution Time
10 million	17	25	2596.858 seconds
10 million	90	25	551.194 seconds
20 million	35	25	2349.266 seconds
20 million	90	25	1122.242 seconds
30 million	54	25	2636.32 seconds
30 million	90	25	1629.504 seconds

Compare with old version

Version	Physical Nodes	Tasks	MSG bytes	MSG exchanges per Superstep	Supersteps	Job Execution Time
Hama 0.4-SNAPSHOT	16	160	16	160000	2048	1060.434 seconds
Hama 0.2-incubating	16	16	16	16000	2048	2030.187 seconds

K-Means Clustering

- Experimental environments
  - One rack (16 nodes 256 cores) cluster
  - 10G network
  - Hadoop 0.20.2
  - Hama 0.4 TRUNK r1222075.
  - Job Execution Time includes the generation of a random dataset of N points and assigning the first k as initial centers.
  - Iterated input vectors on disk and kept centers in main memory.
  - Block partitioning has been used.

N	k	Dimension	Max Iteration	Tasks	Job Execution Time
1 million	10	2	10	1	21.863 seconds
10 million	10	2	10	5	55.339 seconds
10 million	10	2	10	5	45,614 seconds
10 million	10	2	10	20	36,156 seconds
10 million	10	2	10	40	33,199 seconds
100 million	10	2	10	43	81.826 seconds
200 million	10	2	10	85	127.026 seconds
200 million	10	2	10	85	236,682 seconds

http://people.apache.org/~tjungblut/downloads/benchmark_kmeans.png

N	k	Dimension	Max Iteration	Tasks	Job Execution Time
1 million	10	10	20	2	44.56 seconds
10 million	10	10	20	14	69.842 seconds

Single Shortest Path Problem

- Experimental environments
  - One rack (16 nodes 256 cores) cluster
  - 10G network
  - Hadoop 0.20.2
  - Hama 0.4 TRUNK r1213634.
  - MR based Random Graph Generator Random Graph Generator for 0.5.0
  - Task and data partition based on hashing of vertextID in graph and size of input data.

http://4.bp.blogspot.com/-nsYLwF_l3-c/TucnAg0GBCI/AAAAAAAAAU4/VSeUosa2Q58/s1600/scr.png

Vertices (x10 edges)	Tasks	Supersteps	Job Execution Time
10 million	6	5423	656.393 seconds
20 million	12	2231	449.542 seconds
30 million	18	4398	886.845 seconds
40 million	24	5432	1112.912 seconds
50 million	30	10747	2079.262 seconds
60 million	36	8158	1754.935 seconds
70 million	42	20634	4325.141 seconds
80 million	48	14356	3236.194 seconds
90 million	54	11480	2785.996 seconds
100 million	60	7679	2169.528 seconds

Random Communication Benchmark

4 racks

- 1024 VM nodes (1024 cores)
- 10G network

Hama 0.4

- Work in progress

1 rack

- 16 nodes (256 cores)

- 10G network

Hama 0.4 (r.1177507)

- 160 Tasks (10 tasks per node)

Size of each message	Messages per superstep	Number of supersteps	Job runtime
16 bytes	10,000	32	63.875 seconds
16 bytes	20,000	32	81.76 seconds
16 bytes	30,000	32	102.879 seconds
16 bytes	40,000	32	117.783 seconds
16 bytes	50,000	32	129.778 seconds
16 bytes	60,000	32	147.876 seconds
16 bytes	70,000	32	156.896 seconds
16 bytes	80,000	32	184.609 seconds
16 bytes	90,000	32	187.035 seconds
16 bytes	100,000	32	199.027 seconds

- 16 Tasks (1 task per node)

Hama 0.3

Improved compared with 0.2-incubating.

Size of each message	Messages per superstep	Number of supersteps	0.2-incubating	**0.3-incubating**
16 bytes	1000	512	507.837 seconds	486.838 seconds
16 bytes	1000	1024	979.198 seconds	874.016 seconds

Hama 0.2

Test 1 (many small messages vs. few large messages)

Size of each message	Messages per superstep	Number of supersteps	Job runtime
500 bytes	100	16	7.461 seconds
500 bytes	100	32	9.397 seconds
500 bytes	100	64	14.341 seconds
500 bytes	100	128	26.394 seconds
500 bytes	100	256	43.411 seconds
500 bytes	100	512	84.489 seconds
500 bytes	100	1024	156.581 seconds
500 bytes	100	2048	308.671 seconds

Size of each message	Messages per superstep	Number of supersteps	Job runtime
10 kb	5	16	13.679 seconds
10 kb	5	32	23.427 seconds
10 kb	5	64	46.398 seconds
10 kb	5	128	86.476 seconds
10 kb	5	256	171.511 seconds
10 kb	5	512	339.608 seconds
10 kb	5	1024	675.994 seconds
10 kb	5	2048	1872.939 seconds

Test 2

Size of each message	Messages per superstep	Number of supersteps	Job runtime
16 bytes	1000	16	20.365 seconds
16 bytes	1000	32	36.386 seconds
16 bytes	1000	64	67.404 seconds
16 bytes	1000	128	126.503 seconds
16 bytes	1000	256	251.602 seconds

16 bytes	1000	512	507.837 seconds
16 bytes	1000	1024	979.198 seconds
16 bytes	1000	2048	2030.187 seconds