University of A Coruña

Computer Architecture Group

Flame-MR User Guide

Authors:
Jorge Veiga, Roberto R. Expósito,
Guillermo L. Taboada and Juan Touriño

January 31, 2019

Contents

1 Overview 3

2 Environmental variables 3

3 Configuration 3

4 Execution 4

A Hardware affinity 5

B Contact 5

1 Overview

Flame-MR is an in-memory MapReduce framework that improves the performance of Hadoop. It is based on an event-driven architecture that enhances the usage of memory and computing resources, while also applying several optimizations to the overall MapReduce process. One of them is the reduction of memory copies and object creations, which decreases the overhead of the Java garbage collector. Flame-MR also pipelines data movements through network and disk, alternating computation, network utilization and I/O operations. Finally, it applies specific optimization techniques to each MapReduce phase, including in-memory object sort, a k-way merge algorithm and binary comparisons of data fields.

The implementation of Flame-MR is purely based on Java, which ensures its portability among systems. Moreover, it is fully integrated with the Hadoop ecosystem, running on Hadoop YARN and using HDFS for data storage. Flame-MR keeps full compatibility with the Hadoop API, which means that existing MapReduce applications can be executed with Flame-MR without source code modifications.

2 Environmental variables

Flame-MR uses the following environmental variables:

- `FLAMEMR_HOME`: The directory that contains the Flame-MR distribution. It is automatically detected by the executable, so the user does not have to define it.
- `HADOOP_HOME`: The directory that contains the Hadoop distribution.
- `FLAMEMR_CONF_DIR`: The directory that contains the configuration parameters for Flame-MR, which is `$FLAMEMR_HOME/conf` by default.

3 Configuration

The configuration parameters can be established in the `$FLAMEMR_CONF_DIR/flame-mr.conf` file. The main properties are listed below.

- **FlameMR.job.containers.per.node**: Worker containers per node in the cluster. Default: 2.
- **FlameMR.job.container.virtual_cores**: Virtual cores for each worker container. Default: 5.
- **FlameMR.job.container.memory**: Memory size for each worker container. Default: 13500.
- **FlameMR.job.debug**: Enable debug mode. Default: false.
- **FlameMR.job.iterative**: Enable iterative mode to cache intermediate data. Default: false.
- **FlameMR.job.iterative.cache_input**: Enable input data caching in iterative mode. Default: false.
- **FlameMR.merge.outputs**: Number of output partitions generated by merge operations (takes into account key coherence). Default: 1.

- **FlameMR.load.balancing.mode**: Enable load balancing mode to split large reduce input partitions, preventing workload unbalance (does not take into account key coherence). Default: false.
- **FlameMR.load.balancing.threshold**: Maximum partition size in load balancing mode. Default: 0.
- **DataPool.max.percent**: DataPool maximum percentage of the total memory size. Default: 0.7.
- **DataPool.buffer.size**: DataPool buffer size in bytes. Default: 1048576.

4 Execution

The executable file can be found at `$FLAMEMR_HOME/bin/flame-mr`. It can be used to execute the MapReduce application with the same syntax as "hadoop jar".

```
flame-mr hadoop-examples.jar wordcount input output
```

During the execution, the main output of the program will show the running information in a similar way as Hadoop does.

A Hardware affinity

Flame-MR has been prepared to support the use of hardware affinity by using the hwloc tool. By configuring several levels of affinity, Flame-MR is able to bind threads and processes to CPU resources, thus limiting the context changes when executing the data-processing operations. Flame-MR makes use of a Java wrapper for hwloc, jhwloc (available at `https://github.com/rreye/jhwloc`). Hardware affinity can be configured in Flame-MR by means of the following configuration parameters:

- **FlameMR.Worker.affinity.level**: Affinity level to be used in the execution of the workloads. Possible values:
  - NONE: No affinity (default).
  - NUMACTL: CPU affinity by using the Numactl command (no jhwloc/hwloc installation needed).
  - CPU: CPU affinity by using jhwloc.
  - CORE: core affinity by using jhwloc.
  - CORE_SINGLE: core affinity by using jhwloc, using only one PU per core.
  - PU: PU affinity by using jhwloc.
- **FlameMR.Worker.affinity.max**: Maximum number of CPUs to use when using NUMACTL affinity.
- **FlameMR.hwloc.home**: Installation directory of the hwloc tool.
- **FlameMR.jhwloc.home**: Installation directory of the jhwloc tool.

B Contact

Flame-MR has been developed in the Computer Architecture Group at the University of A Coruña by the following authors:

- Jorge Veiga: `http:gac.udc.es/~jveiga`
- Roberto R. Expósito: `http:gac.udc.es/~rreye`
- Guillermo L. Taboada: `http:gac.udc.es/~gltaboada/`
- Juan Touriño: `http:gac.udc.es/~juan`

To report any question, bug, requirement or information about Flame-MR, feel free to contact us (`jorge.veiga@udc.es`).