实验负载

TeraSort 是 Hadoop MapReduce 生态系统中最著名的基准测试程序（Benchmark），曾多次被用于刷新世界排序速度记录。它的核心任务非常简单纯粹：**对大量随机生成的 Key-Value 数据进行全局排序**。

本次实验使用 TeraSort 作为负载，因为它是一个典型的 **I/O 密集型 (I/O Intensive)** 和 **Shuffle 密集型 (Shuffle Heavy)** 任务，能够对集群的 **网络带宽** 和 **磁盘吞吐** 施加巨大压力，是验证 `mapreduce.job.reduce.slowstart.completedmaps` 参数影响的理想场景。

数据生成

TeraSort 处理的数据由配套工具 **TeraGen** 生成，具有严格的格式规范：

- **记录格式：** 每行数据固定为 **100 字节**。
    - **Key (排序键):** 前 **10 字节**，由随机生成的 ASCII 字符组成。
    - **Value (负载值):** 后 **90 字节**，也是随机字符。
- **数据规模：** 本实验分别采用了 1GB、5GB、10GB 和 20GB 的数据集，旨在观察不同压力下的系统表现。

任务负载介绍

1 Map 阶段：采样与局部排序 (Map Phase)

- **采样分区 (TotalOrderPartitioner):** 与普通 MapReduce 任务使用 Hash 取模不同，TeraSort 在作业启动前会对输入数据进行**采样**，构建一个 Trie 树（字典树）。这确保了 Key 的分布是全局有序的（例如：Partition 1 处理 A-C 开头的数据，Partition 2 处理 D-F 开头的数据）。
- **内存排序 (QuickSort):** Mapper 读取数据后，提取 Key，并在内存缓冲区中进行**快速排序 (QuickSort)**。
- **资源特征：** 此阶段主要消耗 **CPU**（用于排序计算）和 **内存带宽**。

2 Shuffle 阶段：数据传输与归并 (The Shuffle Phase)

- 这是本次实验的核心观测点。
- **Copy (拉取):** Reduce 任务启动后，通过网络从各个 Map 节点拉取属于自己 Partition 的数据。`slowstart` 参数直接控制了这个动作的启动时机。
- **Merge (归并):** Reduce 节点在拉取数据的同时，会对溢写到磁盘的多个小文件进行多路归并排序 **(Merge Sort)**。
- **资源特征：** 此阶段是典型的 **网络 I/O** 和 **磁盘 I/O** 密集型。在 0.05 策略下，此阶段与 Map 阶段并行；在 1.0 策略下，此阶段完全串行。

3 Reduce 阶段：最终输出 (Reduce Phase)

- **恒等逻辑 (Identity Reducer):** TeraSort 的 Reducer 逻辑极其简单，不进行任何额外的聚合或计算。因为经过 Shuffle 后，进入 Reducer 的数据流已经是全局有序的。
- **HDFS 写入:** Reducer 的唯一工作就是将有序数据流写入 HDFS。
- **资源特征：** 此阶段主要消耗 **网络带宽**（用于 HDFS 副本复制）和 **磁盘写入**。

环境配置

硬件配置

项目	详细配置	备注
集群拓扑	**3** 节点 (1 Master + 2 Slaves)	物理机集群
CPU 配置	**72 vCores** (每节点)	型号： Intel(R) Xeon(R) Platinum 8347C @ 2.10GHz
内存配置	混合配置 **(总量 > 1.2 TB)**	**Slave1:** 1.0 TB (1000 GB) **Slave2:** 251 GB
网络环境	**1 Gbps (千兆以太网)**	实测峰值吞吐量约 117 MB/s，是本实验的主要瓶颈。
存储介质	NVMe SSD	
总计算资源	**144 vCores / ~1.25 TB** 内存	slave1 : 1007GB ； slave2 ： 256 GB

软件配置

项目	版本信息
操作系统	**Ubuntu 20.04.6 LTS** (Focal Fossa)
内核版本	Linux 5.x
JDK 版本	**OpenJDK 1.8.0_452** (64-Bit Server VM)
Hadoop 版本	**Apache Hadoop 3.3.6**
集群模式	**YARN (Fully Distributed)**

算力过剩，带宽不足”：

> *“本实验集群配备了高性能的* ***Intel Xeon Platinum 8347C (72核)*** *处理器和* ***TB*** *级内存，计算能力极其充裕。然而，节点间仅通过* ***1Gbps*** *千兆网络连接。这种‘强计算、弱网络’的硬件特征，使得 Shuffle 阶段的网络传输成为整个作业的绝对瓶颈，也使得* `Slow Start` *策略的优化效果尤为显著。”*

内存对 I/O 的掩盖：

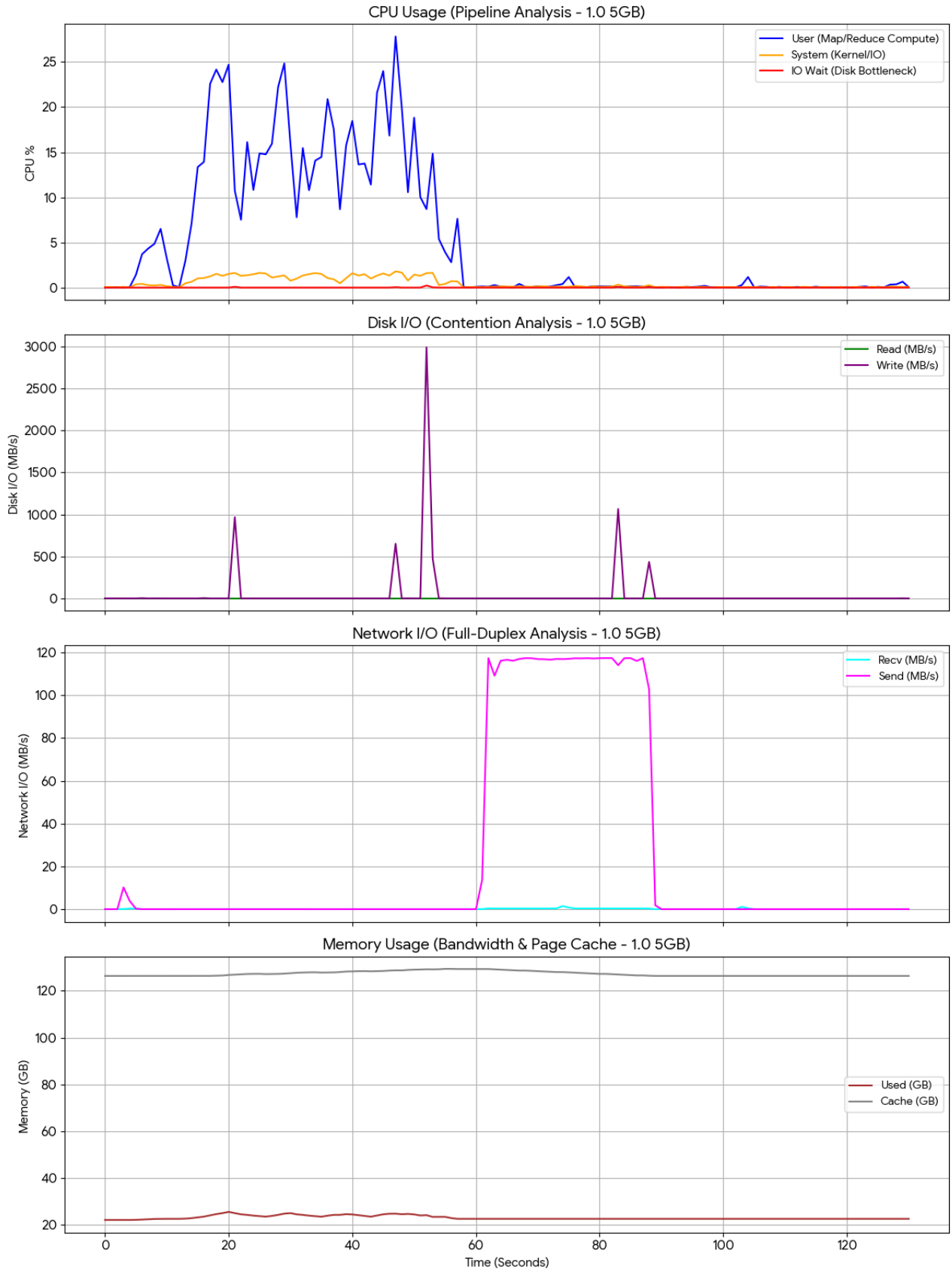
> *“得益于 Slave 节点拥有的超大内存（256GB - 1TB），Linux 操作系统的* ***Page Cache*** *机制有效缓存了绝大部分磁盘读取请求。实验数据显示 Map 阶段的物理磁盘读取（Read I/O）几乎为零，这进一步凸显了网络 I/O 在整体性能中的决定性作用。”*

生成数据

```
# 1. 生成 5GB (5000万行)
hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar teragen 50000000 /input-5g

# 2. 生成 10GB (1亿行)
hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar teragen 100000000 /input-10g

# 3. 生成 20GB (2亿行)
hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar teragen 200000000 /input-20g
```

5G

slow_start = 1.0



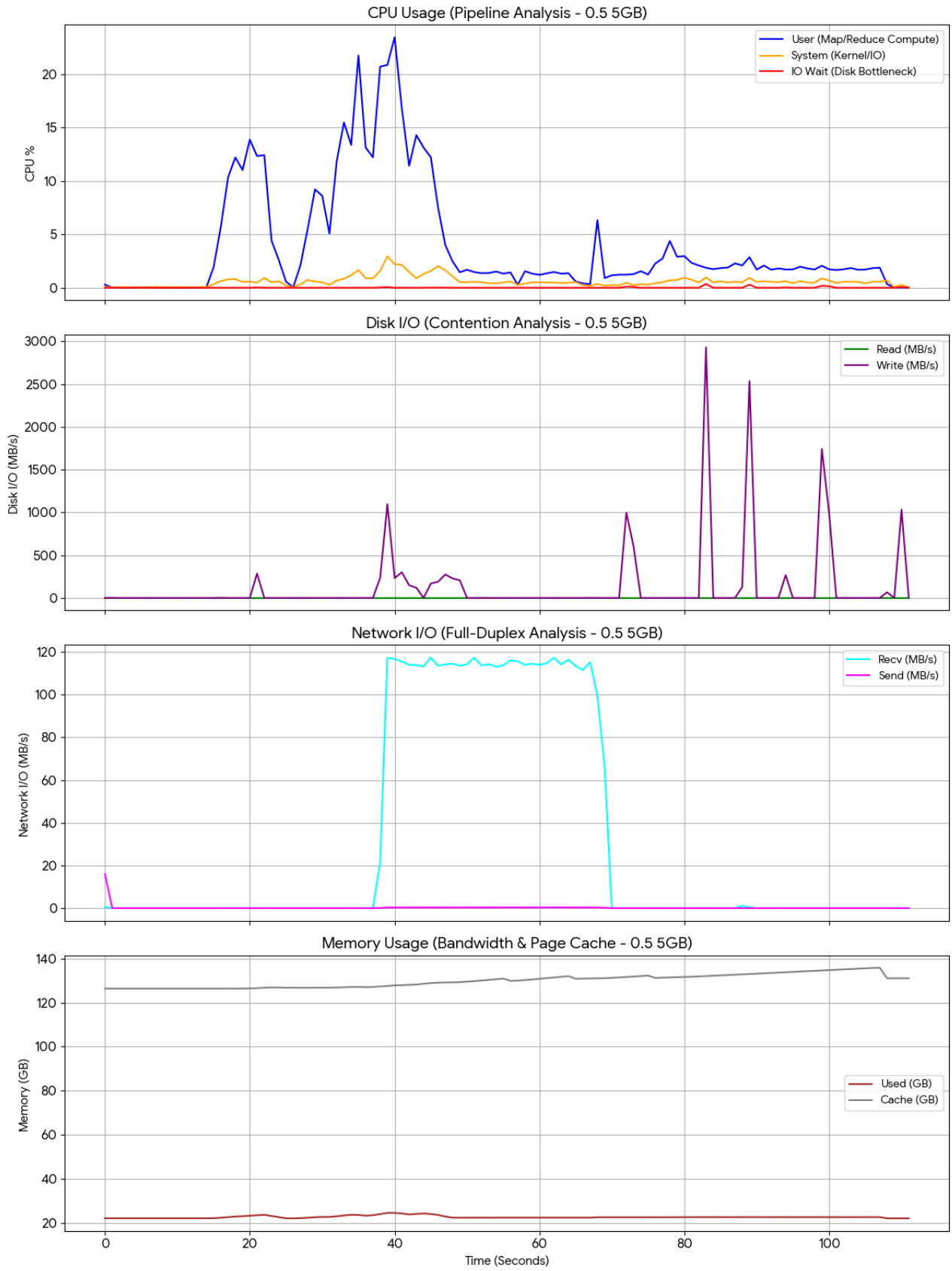
cmd

```
dstat -tcmnd --output 1.0_5g.csv 1

time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=1.0 /input-5g
/output-5g-1.0
```

res

```
2025-11-29 00:52:40,392 INFO terasort.TeraSort: done

real    2m8.414s
user    0m13.319s
sys 0m1.175s
```

slow_start = 0.5


CPU Usage (Pipeline Analysis - 0.5 5GB)
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)
CPU %
Disk I/O (Contention Analysis - 0.5 5GB)
Read (MB/s)
Write (MB/s)
Disk I/O (MB/s)
Network I/O (Full-Duplex Analysis - 0.5 5GB)
Recv (MB/s)
Send (MB/s)
Network I/O (MB/s)
Memory Usage (Bandwidth & Page Cache - 0.5 5GB)
Used (GB)
Cache (GB)
Memory (GB)
Time (Seconds)

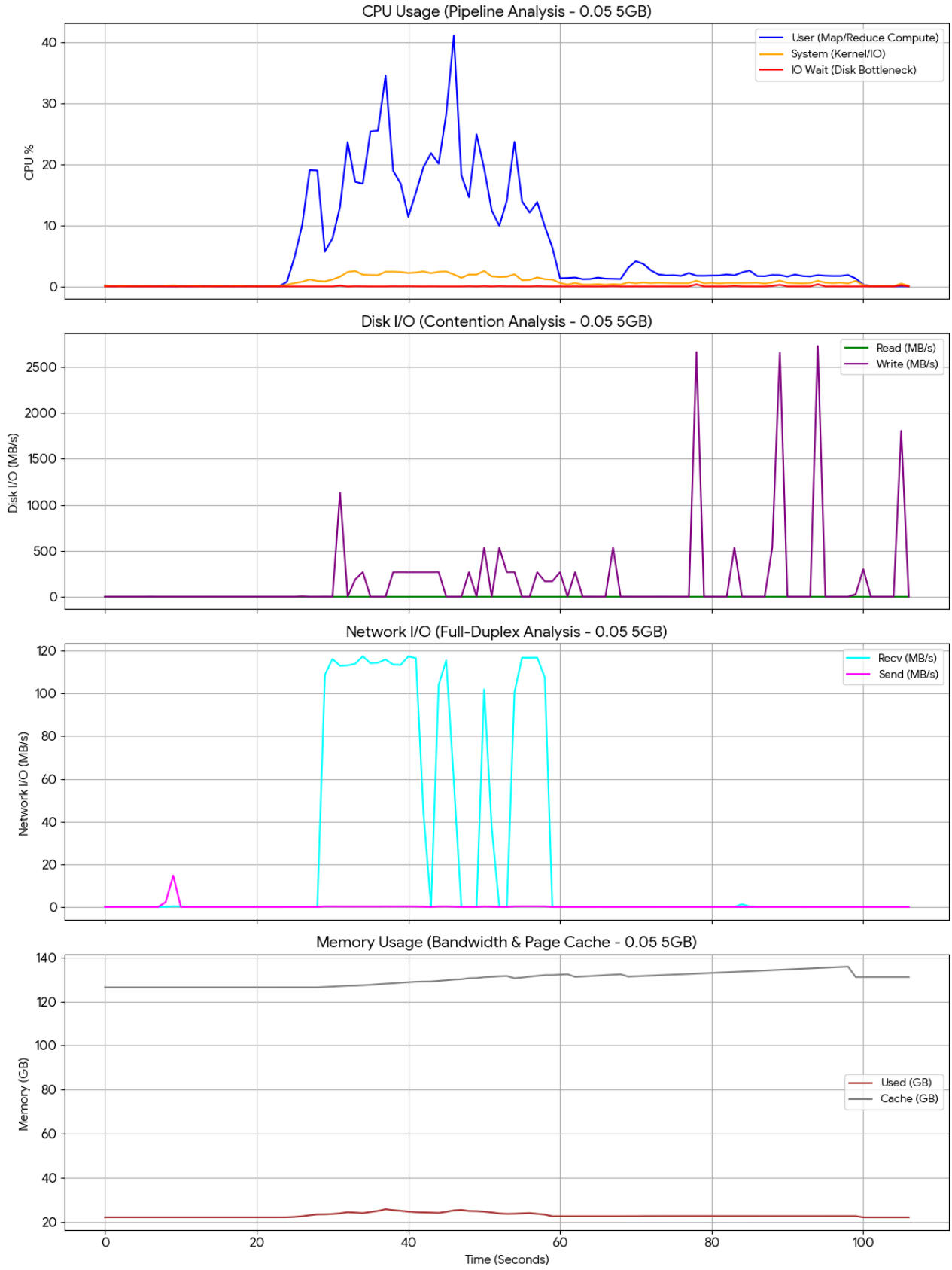
cmd

```
# 生成监控日志
dstat -tcmnd --output 0.5_5g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.5 /input-5g
/output-5g-0.5
```

res

```
2025-11-29 00:58:09,701 INFO terasort.TeraSort: done

real    1m52.376s
user    0m11.920s
sys 0m0.966s
```

slow_start = 0.05


CPU Usage (Pipeline Analysis - 0.05 5GB)
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)
CPU %
Disk I/O (Contention Analysis - 0.05 5GB)
Read (MB/s)
Write (MB/s)
Disk I/O (MB/s)
Network I/O (Full-Duplex Analysis - 0.05 5GB)
Recv (MB/s)
Send (MB/s)
Network I/O (MB/s)
Memory Usage (Bandwidth & Page Cache - 0.05 5GB)
Used (GB)
Cache (GB)
Memory (GB)
Time (Seconds)

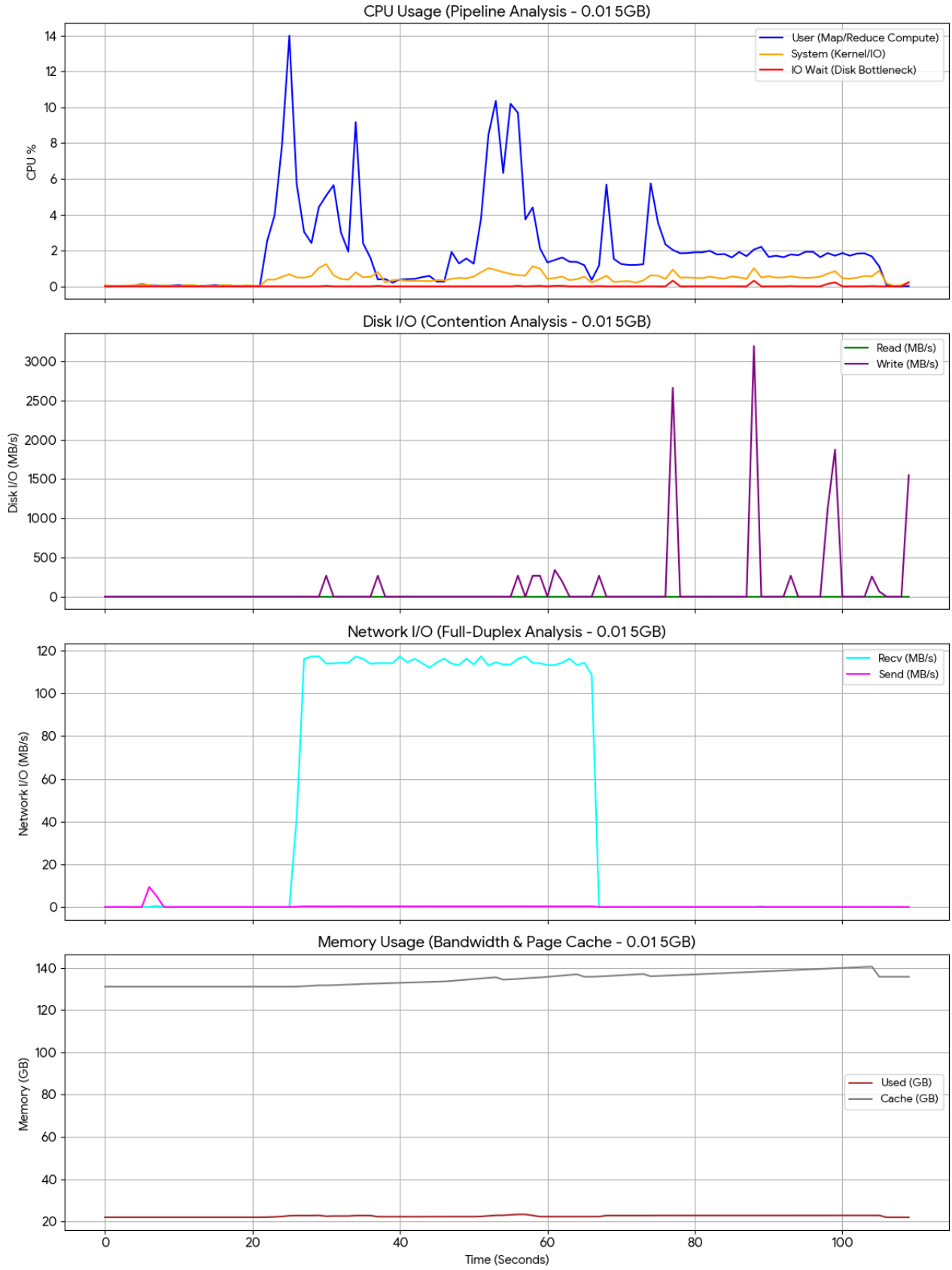
cmd

```
dstat -tcmnd --output 0.05_5g.csv 1

time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.05 \
/input-5g /output-5g-0.05
```

res

```
real    1m35.879s
user    0m12.766s
sys 0m1.043s
```

slow_start = 0.01


CPU Usage (Pipeline Analysis - 0.01 5GB)
CPU %
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)
Disk I/O (Contention Analysis - 0.01 5GB)
Disk I/O (MB/s)
Read (MB/s)
Write (MB/s)
Network I/O (Full-Duplex Analysis - 0.01 5GB)
Network I/O (MB/s)
Recv (MB/s)
Send (MB/s)
Memory Usage (Bandwidth & Page Cache - 0.01 5GB)
Memory (GB)
Used (GB)
Cache (GB)
Time (Seconds)

cmd

```
# 生成监控日志
dstat -tcmnd --output 0.01_5g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.01 /input-5g
/output-5g-0.01
```

res

```
real    1m43.743s
user    0m12.344s
sys 0m0.953s
```

10GB

---

slow_start = 0.01


CPU Usage (Pipeline Analysis - 0.01 10GB)
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)
CPU %
Disk I/O (Contention Analysis - 0.01 10GB)
Read (MB/s)
Write (MB/s)
Disk I/O (MB/s)
Network I/O (Full-Duplex Analysis - 0.01 10GB)
Recv (MB/s)
Send (MB/s)
Network I/O (MB/s)
Memory Usage (Bandwidth & Page Cache - 0.01 10GB)
Used (GB)
Cache (GB)
Memory (GB)
Time (Seconds)

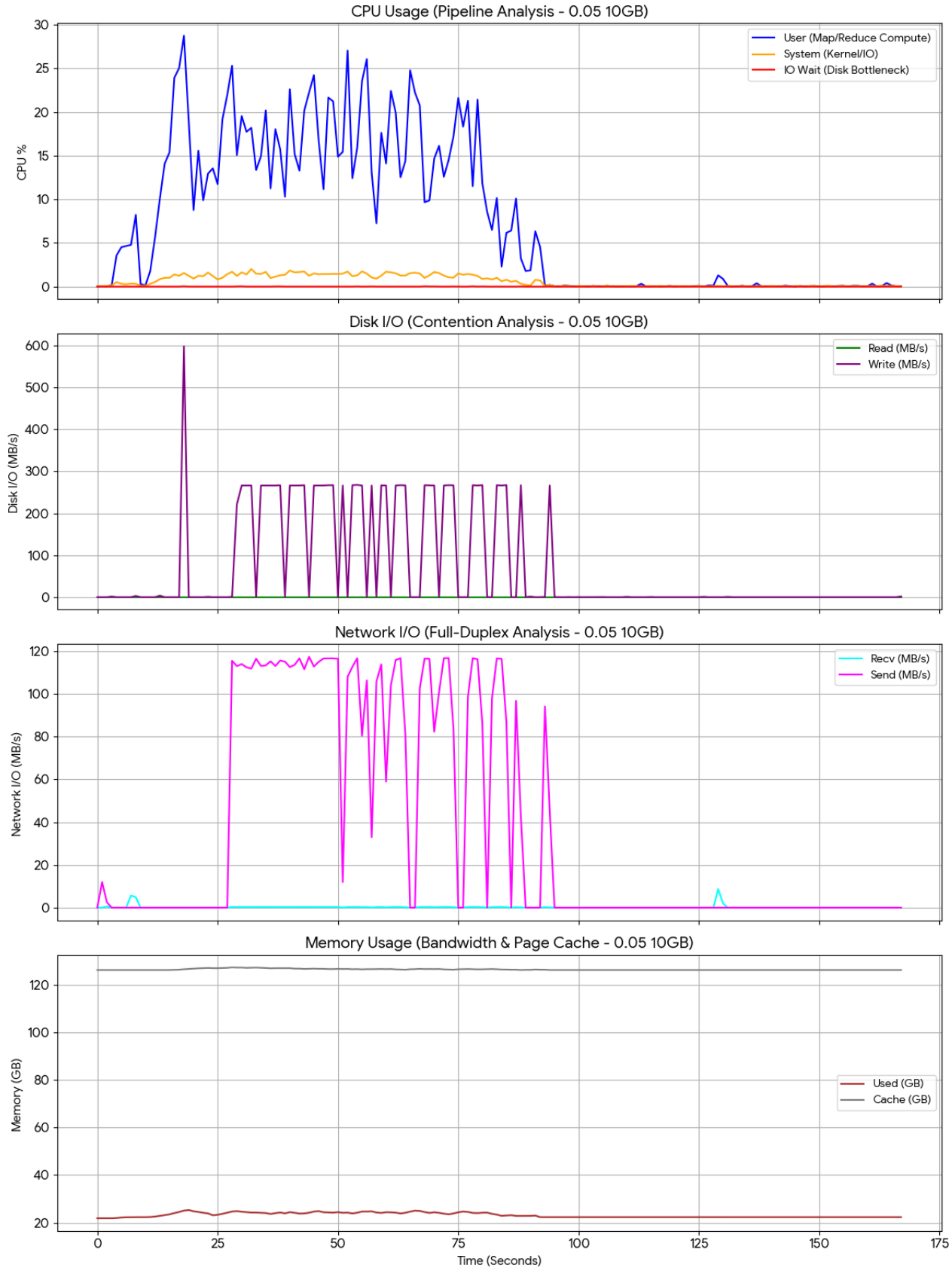
cmd

```
# 生成监控日志
dstat -tcmnd --output 0.01_10g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.01 /input-10g
/output-10g-0.01
```

res

```

real    2m47.522s
user    0m14.315s
sys 0m1.295s
```

slow_start = 0.05


CPU Usage (Pipeline Analysis - 0.05 10GB)
CPU %
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)
Disk I/O (Contention Analysis - 0.05 10GB)
Disk I/O (MB/s)
Read (MB/s)
Write (MB/s)
Network I/O (Full-Duplex Analysis - 0.05 10GB)
Network I/O (MB/s)
Recv (MB/s)
Send (MB/s)
Memory Usage (Bandwidth & Page Cache - 0.05 10GB)
Memory (GB)
Used (GB)
Cache (GB)
Time (Seconds)

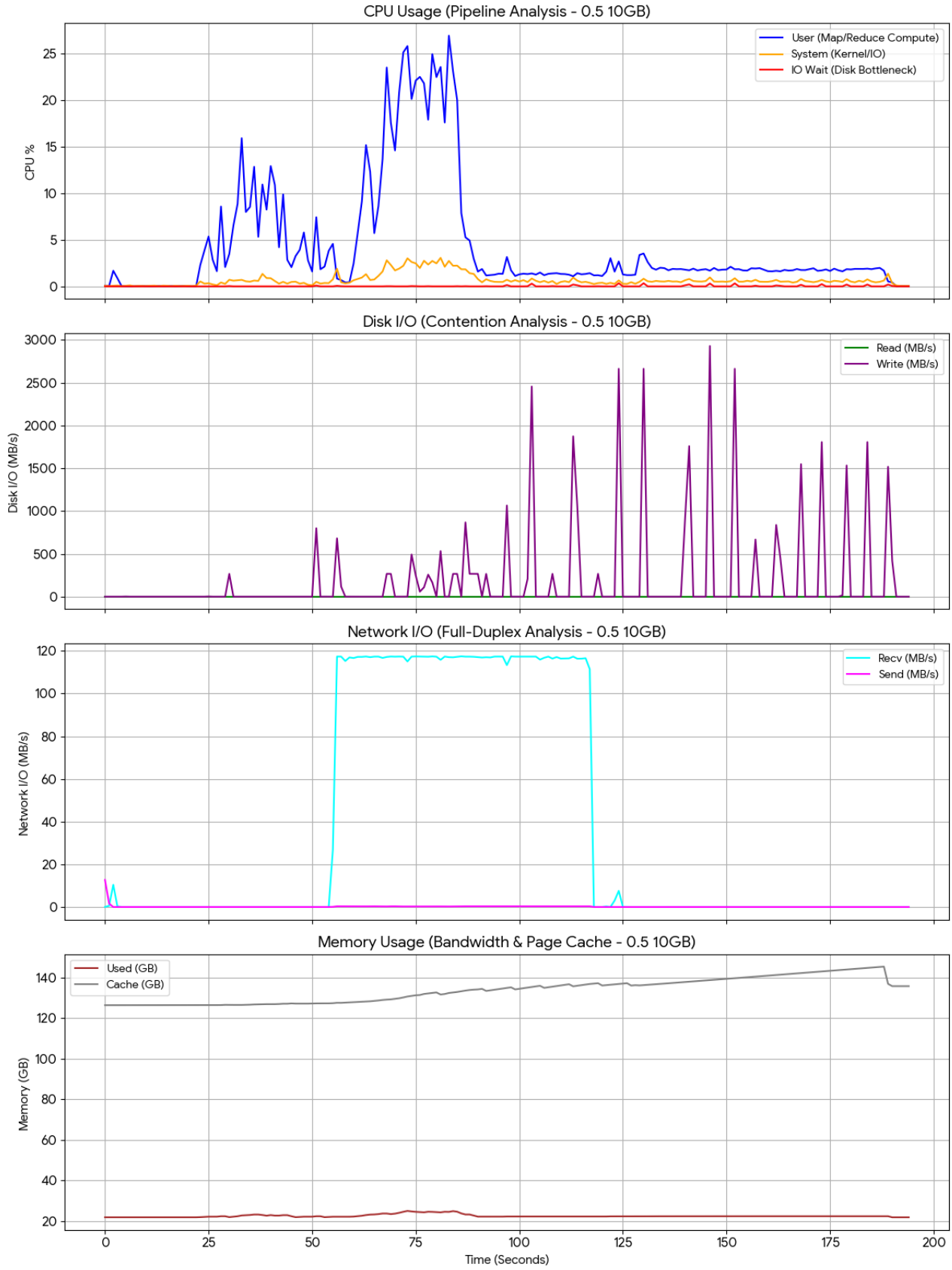
cmd

```
# 生成监控日志
dstat -tcmnd --output 0.05_10g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.05 /input-10g
/output-10g-0.05
```

res

```
real    2m46.644s
user    0m13.291s
sys 0m1.460s
```

slow_start = 0.5


CPU Usage (Pipeline Analysis - 0.5 10GB)
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)
CPU %
Disk I/O (Contention Analysis - 0.5 10GB)
Read (MB/s)
Write (MB/s)
Disk I/O (MB/s)
Network I/O (Full-Duplex Analysis - 0.5 10GB)
Recv (MB/s)
Send (MB/s)
Network I/O (MB/s)
Memory Usage (Bandwidth & Page Cache - 0.5 10GB)
Used (GB)
Cache (GB)
Memory (GB)
Time (Seconds)

cmd

```
# 生成监控日志
dstat -tcmnd --output 0.5_10g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.5 /input-10g
/output-10g-0.5
```

res

```
real    3m15.502s
user    0m14.301s
sys 0m1.214s
```

slow_start = 1.0


CPU Usage (Pipeline Analysis - 1.0 10GB)
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)
CPU %
Disk I/O (Contention Analysis - 1.0 10GB)
Read (MB/s)
Write (MB/s)
Disk I/O (MB/s)
Network I/O (Full-Duplex Analysis - 1.0 10GB)
Recv (MB/s)
Send (MB/s)
Network I/O (MB/s)
Memory Usage (Bandwidth & Page Cache - 1.0 10GB)
Used (GB)
Cache (GB)
Memory (GB)
Time (Seconds)

cmd

```
# 生成监控日志
dstat -tcmnd --output 1.0_10g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=1.0 /input-10g
/output-10g-1.0
```

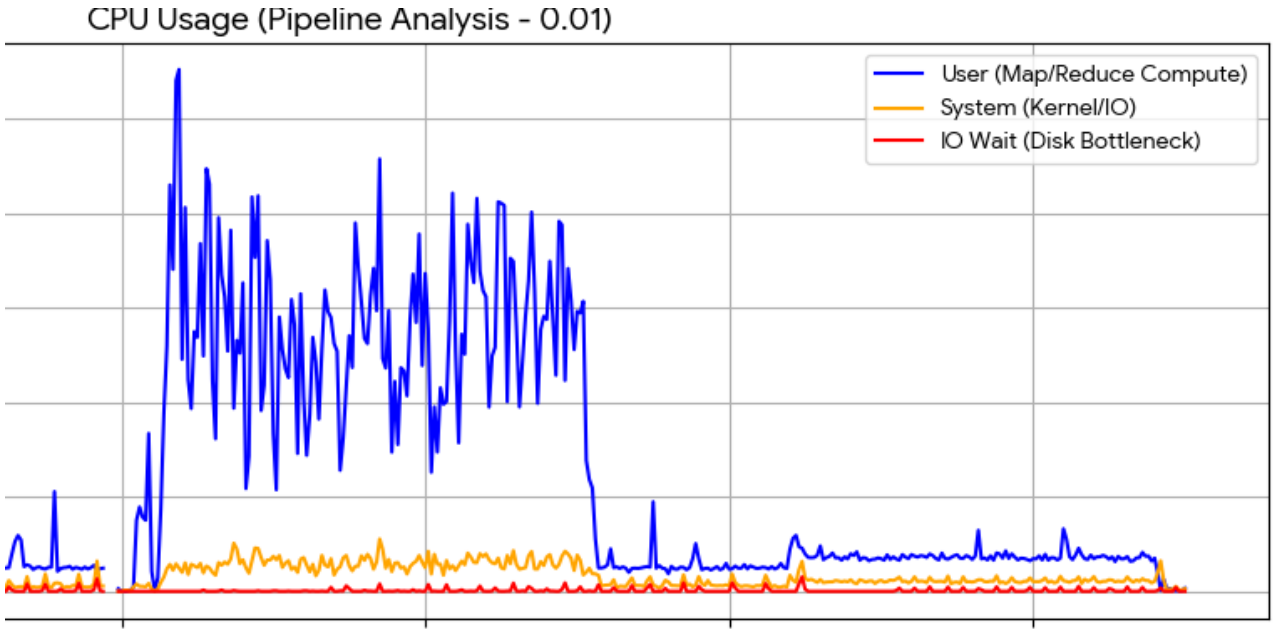
res

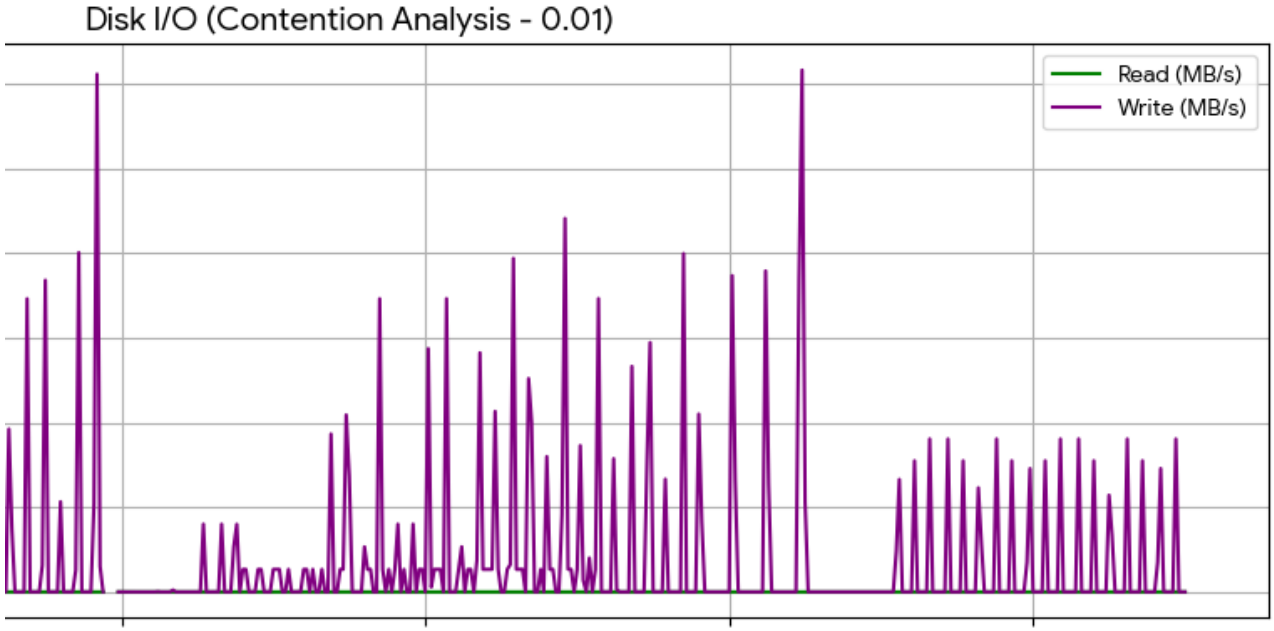
```
real    3m22.811s
user    0m11.983s
sys 0m1.115s
```

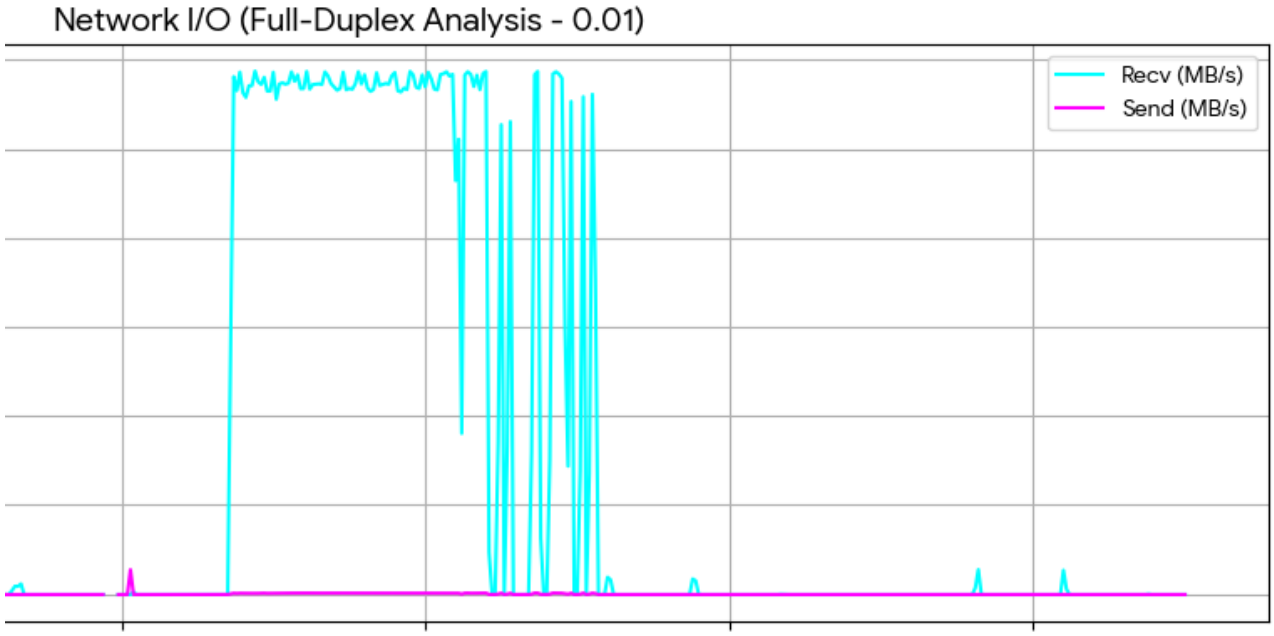
20GB

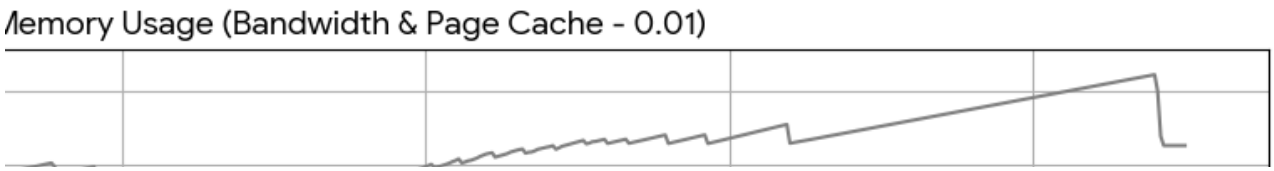
---

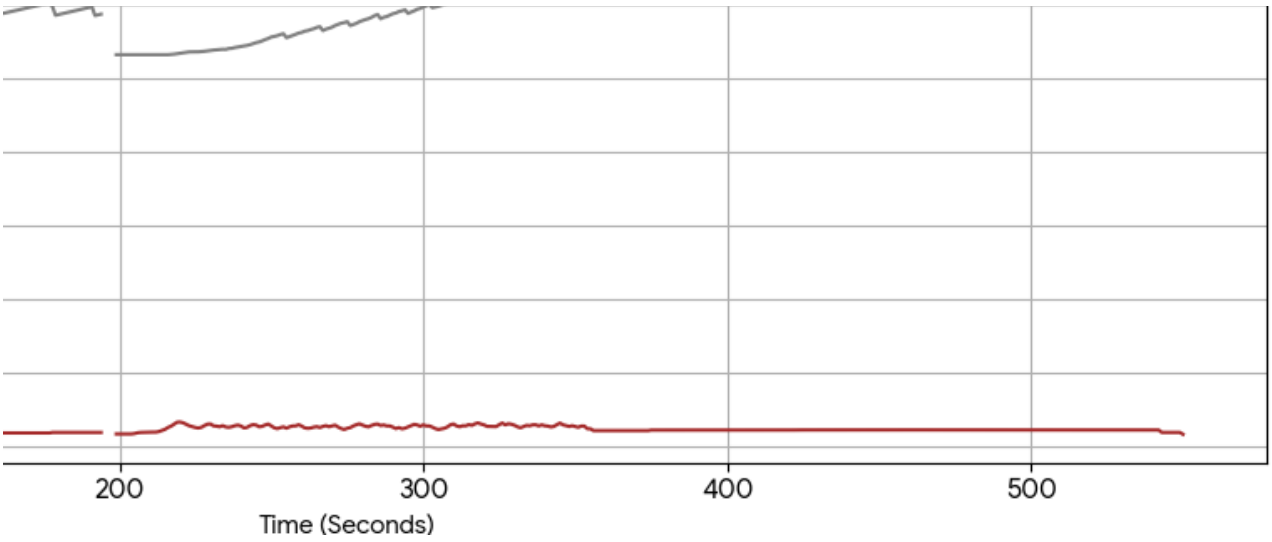
slow_start = 0.01


CPU Usage (Pipeline Analysis - 0.01)
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)


Disk I/O (Contention Analysis - 0.01)
Read (MB/s)
Write (MB/s)


Network I/O (Full-Duplex Analysis - 0.01)
Recv (MB/s)
Send (MB/s)


Memory Usage (Bandwidth & Page Cache - 0.01)



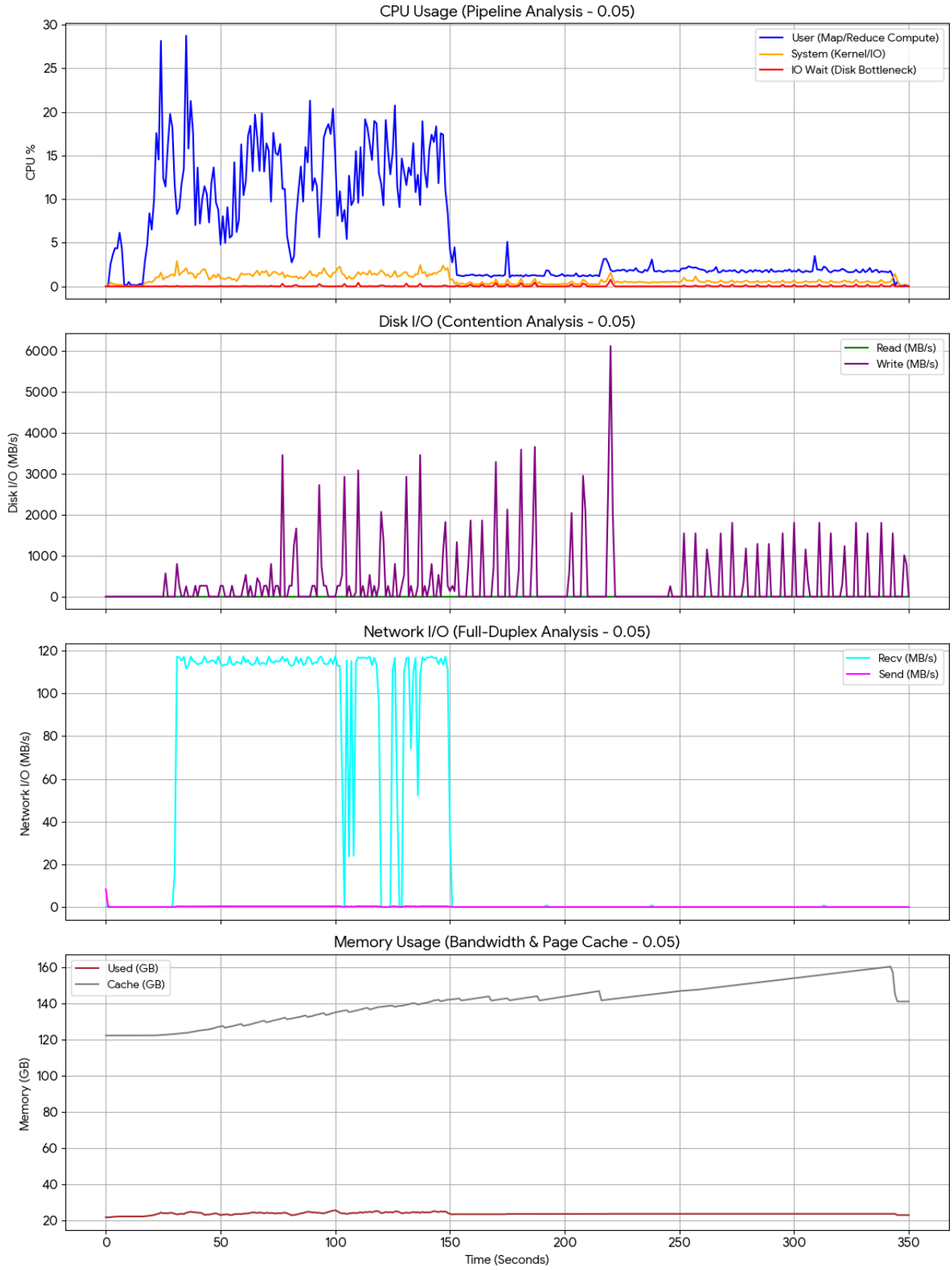
cmd

```
# 生成监控日志
dstat -tcmnd --output 0.01_20g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.01 /input-20g /output-20g-0.01
```

res

```
real    5m28.653s
user    0m13.638s
sys 0m1.318s
```

slow_start = 0.05


CPU Usage (Pipeline Analysis - 0.05)
User (Map/Reduce Compute)
System (Kernel/IO)
IO Wait (Disk Bottleneck)
CPU %
Disk I/O (Contention Analysis - 0.05)
Read (MB/s)
Write (MB/s)
Disk I/O (MB/s)
Network I/O (Full-Duplex Analysis - 0.05)
Recv (MB/s)
Send (MB/s)
Network I/O (MB/s)
Memory Usage (Bandwidth & Page Cache - 0.05)
Used (GB)
Cache (GB)
Memory (GB)
Time (Seconds)

cmd

```
dstat -tcmnd --output 0.05_20g.csv 1

time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.05 /input-20g
/output-20g-0.05
```

res

```
real    5m47.101s
user    0m14.751s
sys 0m1.342s
```

slow_start = 0.5

cmd

```
# 生成监控日志
dstat -tcmnd --output 0.5_20g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=0.5 /input-20g
/output-20g-0.5
```

res

```
real    6m17.420s
user    0m14.964s
sys 0m1.427s
```

slow_start = 1.0

cmd

```
# 生成监控日志
dstat -tcmnd --output 1.0_20g.csv 1

# 运行terasort
time hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar terasort \
-Dmapreduce.job.reduce.slowstart.completedmaps=1.0 /input-20g /output-20g-1.0
```

res

```
real    7m19.140s
user    0m14.685s
sys 0m1.600s
```