计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(12):233−239 [doi: 10.15888/j.cnki.csa.006111]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

# 卫星群分布式技术的研究与实现①

侯玉峰, 倪　明

(华东计算技术研究所 航天产品部, 上海 201800)

**摘　要**: 随着航天技术的发展, 越来越多的信息需在星上处理, 卫星群分布式技术已成为近年来研究的热点. 与地面环境不同, 卫星受到体积、功耗、空间辐射等条件的限制, 在卫星群建立分布式环境需掌握架构设计、操作系统、资源管理等关键技术. 针对以上问题, 本文先研究卫星群分布式计算与存储架构、其次研究卫星群分布式资源监控的实现过程, 最后通过卫星群验证系统证明其可行性与优越性.

**关键词**: 卫星; 分布式; 计算; 存储; 资源监控

引用格式: 侯玉峰,倪明.卫星群分布式技术的研究与实现.计算机系统应用,2017,26(12):233–239. http://www.c-s-a.org.cn/1003-3254/6111.html

## Research and Realization of Distributed Technology on Satellite Cluster

HOU Yu-Feng, NI Ming

(Department of Aerospace Products, East China Institute of Computer Technology, Shanghai 201800, China)

**Abstract**: With the development of space technology, more and more information needs to be computed on the satellite. The distributed technology of satellite cluster hence has become a research hotspot in recent years. Different from the ground environment, the satellite is limited with some constraints such as volume, power consumption and space radiation. The key technology such as architecture design, operating systems and resource management should be grasped to establish distributed environment on satellite cluster. To solve problems above, this paper first studies the distributed computing architecture and storage architecture on satellite cluster. Then, it focuses on the implementation of distributed resource monitoring on satellite cluster. Finally, it proves its feasibility and superiority through the authentication system of satellite cluster.

**Key words**: satellite; distributed; computing; storage; resource monitoring

## 1　引言

随着信息技术的发展, 人们已进入了大数据时代, 构建集群对海量数据进行分布式计算与存储已成为一种共识, 航天卫星领域也同样如此. 构建卫星群分布式架构很有意义, 在国家层面上, 面对纷乱复杂的周边形势, 若能实现卫星组网进行分布式计算与存储, 便可大幅提高卫星能力, 扩展卫星功能, 对国防侦查、态势感知、地图导航、通信指挥等起到更大作用. 在公众层面上, 人们可通过移动设备随时接入卫星网络, 从而享受卫星群系统提供的更加精确的实时定位、物流追踪和遥感信息服务. 综上, 通过卫星组网进行分布式计算与存储已成为航天卫星领域发展的必然趋势[1].

当下地面环境上的分布式计算技术已相当成熟, 但由于卫星所处太空环境的限制, 计算机需要高标准的制造要求、低功耗的处理架构, 以及剪裁过的操作系统, 因而卫星群分布式环境的构建与地面有明显差异[2].

## 2　卫星群分布式架构

针对引言所述问题, 本文选择了八台采用 ARM 体系架构, 搭载银河麒麟操作系统的国产飞腾服务器作

① 收稿时间: 2017-03-14; 修改时间: 2017-03-31; 采用时间: 2017-04-07

为卫星计算机. 整个集群系统分为天地两部分, 地面共三台服务器, 通过电口交换机互连, 其中一台模拟管控台计算机, 通过搭建 django 框架, 编写 web 应用程序显示卫星群的信息, 剩余两台采用飞腾的存储服务器用于模拟数据中心. 天上采用五台剪裁过的低功耗飞腾服务器来模拟卫星, 服务器间通过光口交换机互连, 模拟卫星群的点对点激光通信过程, 天地集群间通过光纤收发器的光电转化技术进行互连. 集群系统的总体设计架构如图 1 所示.

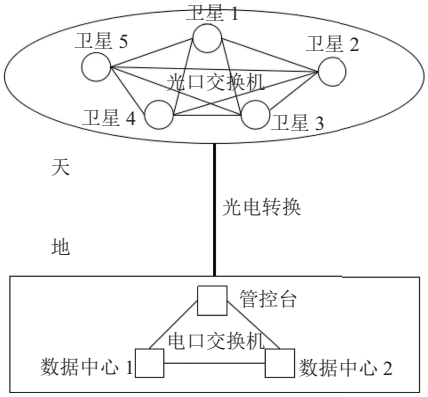


图 1 卫星群总体设计架构

卫星群的分布式架构分为存储和计算两部分, 分别在 2.1 节和 2.2 节中介绍.

### 2.1 卫星群分布式存储架构

卫星群中的一台节点记作 master 节点, 剩余的四台记作 slave 节点, 各节点的本地文件系统, 在逻辑上是一个统一的存储系统. 当某台卫星进行存储时, 会根据冗余备份机制, 选择网络距离最近的两个卫星进行存储, 保持数据的备份数为 3. 进行存储时, master 节点运行 namenode 进程, 负责管理卫星文件系统的命名空间, 维护文件系统树, 保存命名空间镜像文件和编辑日志文件, 此外还记录每个文件中各个块的节点信息. slave 节点运行 datanode 进程, 是存储数据块的工作节点, 并定期向 master 节点发送它们所存储的块列表[3].

卫星群采用分布式存储架构后, 不仅能够增大存储空间, 而且可以将监测到的数据进行冗余备份, 增强了数据的安全性, 存储备份过程如图 2, 箭头指向为数据备份的目的地.

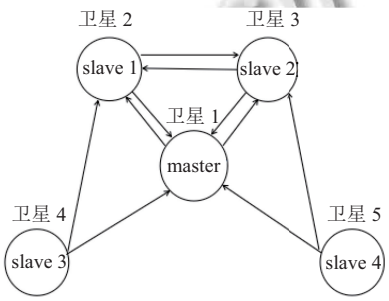


图 2 卫星群存储备份示意图

### 2.2 卫星群分布式计算架构

master 节点从卫星群的文件系统取数据后, 按照指定的大小分成多个 block, 接着将 block 切分成多个 inputsplit 传送到各 slave 节点处. 然后各个 slave 节点通过 map 程序对块进行处理, 结果通过 combine 函数预处理后先写入环缓冲, 缓冲区溢满后, 再将结果进行分区、排序、写磁盘操作. 随后 master 节点通过 Reduce 程序从卫星群的文件系统中取到相应的数据块进行处理, 最后将所有 Reduce 程序的处理结果存储到卫星群文件系统中[4-6].

卫星群采用分布式计算架构后, 通过对数据进行分发和归约操作使计算速度加快, 计算流程如图 3.

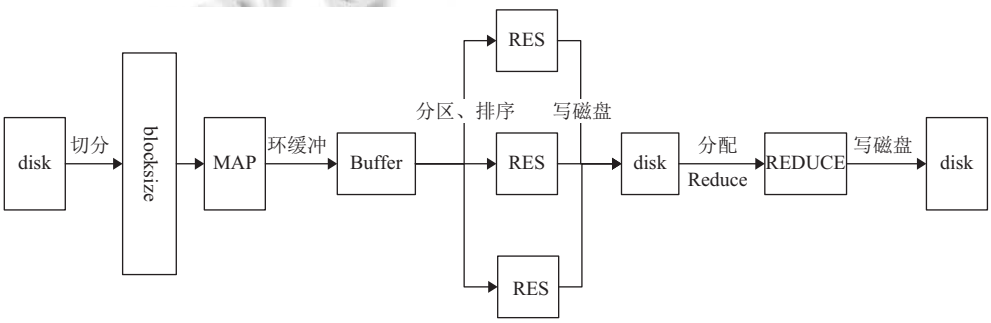


图 3 卫星群分布式计算流程图

## 3 卫星群的分布式资源监控

资源监控是集群部署中的关键组件, 具有实时、精确、全面的特性, 通过采集集群运行指标, 展示详细的工作状态. 卫星群部署资源监控系统, 一方面是为了

保证卫星群运行稳定, 让运维人员能第一时间发现节点宕机、失连等故障, 及时解决问题, 从而将损失降到最低. 另一方面也是评判星上应用程序是否高效的一个重要依据. 卫星群的资源监控运用了 Ganglia 的原理, 所有需要监控的卫星节点上运行 gmond 进程, 用于收集卫星信息, 并将这些数据以 XML 格式定时发送给主节点. 主节点运行 gmetad 进程, 接收各从节点的数据, 并以 rrdtool 的形式存放在 rrds 文件中. 与此同时, 主节点运行 gweb 进程, 该进程通过 HTML、PHP、JS 等前端技术, 读取监控数据文件, 并将其显示到浏览器中.

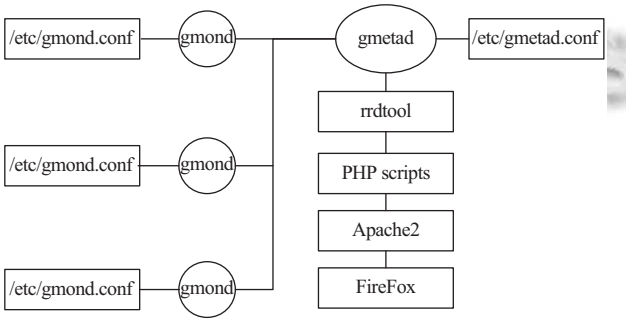


图 4 卫星群资源监控数据流图

执行流程如图 4 所示, 首先修改 gmond 与 gmetad 的配置文件, 包括设置卫星群信息、存储信息以及采样频率. 配置设置好后, 启动 gmond 与 gmetad 进程, 开始监控集群. 集群信息每隔设定时间存储到 rrdtool 中, 对应于 master 节点的 rrds 目录. 然后通过 PHP 文件获取卫星群信息, TPL 和 JS 文件进行界面的布局, Apache2 服务器和 FireFox 浏览器进行代码解析和前端显示. 通过使用分布式资源监控技术, 监控卫星群的运行数据, 从而可对卫星群进行性能检测、故障维修以及资源分配.

### 3.1 监控指标

卫星群的监控指标包括 CPU 使用率、MEMORY 使用率、LOAD 负载率、NET 负载率. 数据的计算方法如下.

(1) CPU 使用率

CPU 有内核态、用户态和空闲态三种状态. 用 $C_c$ 表示内核态, $C_u$ 表示用户态, $C_i$ 表示空闲态. 通过命令 vmstat 的返回值, 获取卫星的这三个数据值后, CPU 使用率 $CPU_u$ 表达式如下:

$$CPU_u = ((C_c + C_u)/(C_c + C_u + C_i))*100\%$$

(2) 内存空闲率

用 $M_f$ 表示空闲内存量, $M_t$ 表示总内存量. 通过读取内存文件获取 free 属性值后, 内存空闲率 $MEM_i$ 表达式如下:

$$MEM_i = (M_f/M_t)*100\%$$

(3) LOAD 负载率

进程有运行态、就绪态和阻塞态三种状态. 用 $P_e$ 表示运行进程数, $P_r$ 表示就绪进程数, $P_b$ 表示阻塞进程数. 通过读取进程文件获取 ps 各属性值后, 平均负载率 $PRO_r$ 表达式如下:

$$PRO_r = ((P_e)/(P_e + P_r + P_b))*100\%$$

(4) 网络负载率

从本星流入流量记作 $N_i$, 从本星流出流量记作 $N_o$, 本星网络吞吐能力记作 $N_t$, 网络负载率 $NET_l$ 表达式如下:

$$NET_l = ((N_i+N_o)/N_t)*100\%$$

### 3.2 存储结构

卫星群的数据存储结构为一个四维数组. 以 CPU 的度量数据为例, 第一维包括 user、nice、system、wait、idea 五项指标. 每项指标下有各类信息, 其中 datapoint 存储一段时间的数据, 共 241 个采样时间点, 每个采样时间点存储 (数据值, 时间) 的键值对. 如图 5 所示.

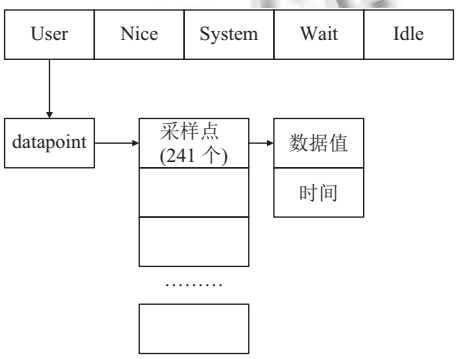


图 5 卫星群 CPU 数据结构

### 3.3 运行机制

心跳监测是实现卫星群资源监控的运行机制, 其中卫星主节点运行 RecourseManager 进程, 卫星从节点运行 NodeManager 进程. 从节点运行一个循环, 每隔一段时间向主节点发送心跳 Heartbeat, 在此期间, 将从节点采集的数据传送给主节点[5]. 主节点接受到心跳的同时, 将收集到的数据存入 rrdtool 中, 并发出 reply. 发送

心跳的函数为 transmitHeartBeat(), 接受心跳的函数为 heartbeat() 函数, 如图 6 所示.

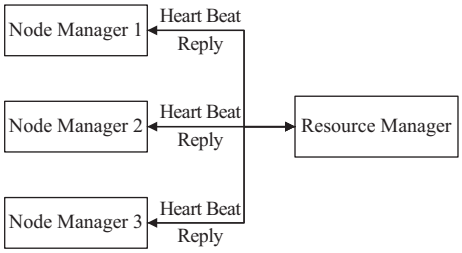


图 6 心跳监测机制图

### 3.4 核心代码

卫星群资源监控底层代码通过调用 gmetad() 方法获取数据, 使用 socket 与卫星主节点的 8652 端口进行交互, 将卫星群各节点的性能指标, 以 xml 数据格式存储于 rrds 文件夹内[7]. 核心代码如下:

```
switch ($context){
    case “cluster”:
        xml_set_element_handler($parser, “start_cluster”, “end_all”);
        $request = “/$clustername”;
        break;
    case “cluster-summary”:
        xml_set_element_handler($parser, “start_cluster_summary”, “end_all”);
        $request = “/$clustername?filter=summary”;
        break;
    case “host”:
        xml_set_element_handler($parser, “start_host”, “end_all”);
        $request = “/$clustername/$hostname”;
        break;
}
```

### 3.5 改进与完善

卫星群原有的监控界面简陋, 监控指标少, 且只能进行一段时间的整体监控, 如图 7.

本人在熟悉监控代码的基础上, 利用 js、php、css 等技术, 完成了对图表样式以及界面的改进、并增添了卫星实时数据动态监控、卫星节点数据对比、卫星节点动态部署等功能. 主要实现过程如下.

(1) 图表样式美化

在 test_cluster.js 文件中, 利用 js 的 jquery 技术获取了卫星群的信息, 之后调用 highcharts 图表库编写绘图函数 draw_line、draw_area、draw_histo、draw_pie, 实现了图表样式的美化[8]. 数据获取函数如下:

```
series: (function(){
    var obja = [];
    for(var i=0; i<b.length; i+=1){
        var obj = {};
        obj[‘name’] = b[i][‘metric_name’];
        var data = [];
        var atlen =b[0][‘datapoints’].length;
        for(var j=0; j<atlen; j+=1){
            data.push(b[i][‘datapoints’][j][0]);
        }
        obj[‘data’] = data;
        obja.push(obj);
    }
    return obja;
}
```

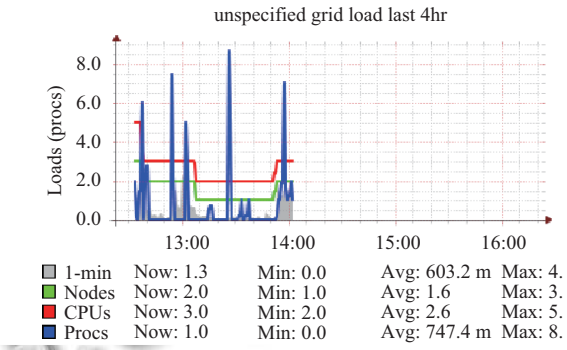


图 7 修改前监控图表

效果图如图 8.

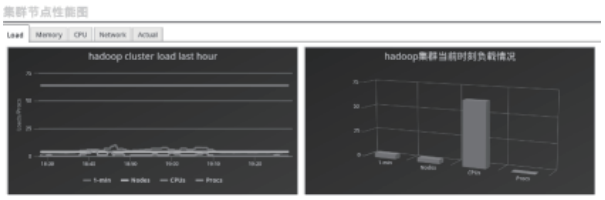


图 8 修改后监控图表

(2) 节点数据对比功能

通过编写节点对比函数 draw_total, 实现多个卫星节点的数据汇总. 在此图表内, 可对比卫星主节点 master 与任意卫星从节点 slave 的数据, 以便进行卫星节点间的性能分析与评估[9]. 部分实现代码如下:

```
$.get("http://"+ipAddress+"/hadoop/graph.php?c="+cur_c+"&h="+namenode+", function(data){
        b1 = data;
$.get("http://"+ipAddress+"/hadoop/graph.php?c="+cur_c+"&h="+datanode1+", function(data){
        b2 = data;
        setoptions();
        b3 = [];
        b3.push(b1[0]);
        b3.push(b2[0]);
        b3.push(b3[0]);
        drawtotal(b3, cur_c, cur_r, "load_one", cur_st);
        });
    });
});
```

效果图如图 9.

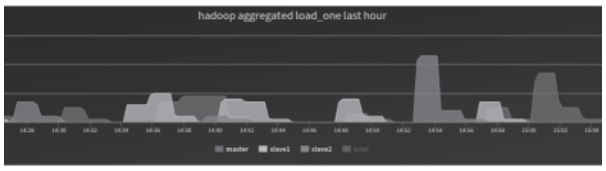


图 9 节点对比图

(3) 实时数据动态监控

test_dynamic.js 文件用于动态显示卫星群的实时数据, 具体过程见下.

打开受控节点的 gmond.conf 文件, 参数 collect_every 值为采样频率, 设置为 1, time_threshold 值为时间阈值, 设置为 300. 之后设置数据存储大小, 修改配置语句"RRA:AVERAGE:0.5.1.1:244", 该配置语句表示存储空间默认每 15s 存一次数据, 共存够 244 个后. 由于采样频率已改为 1, 固须将存储空间加大, 将 244 修改为 3000 即可. 随后将主节点下的 rrds 文件夹删除, 重启卫星群后, 修改采样间隔操作即可成功. 最后在 test_dynamic.js 文件中开启 highcharts 的动画属性 animation, 开启方法如下:

```
chart: {
    type: 'spline',
    animation: Highcharts.svg,
},
```

动态监控效果图如图 10.

(4) 动态增减节点

为实现卫星群的动态部署, 需添加卫星节点的动态增减功能. 可通过修改 hdfs-sites 文件, 将节点状态从 live 转换为 decommissioned, 表示所选卫星节点已与卫星群断开连接. 相关配置如下:

```
<property>
<name>dfs.hosts.exclude</name>
<value>/etc/hadoop/exclude</value>
</property>
```


图 10 实时数据动态显示

## 4 卫星群验证系统

### 4.1 硬件配置

卫星硬件配置如表 1 所示.

表 1 卫星硬件配置

| 名称 | 数量 | 详细 |
|---|---|---|
| 处理器 | 16个 | FT1500(1.5 GHZ) |
| 主板 | 1个 | 飞腾 |
| 内存 | 1个 | 32 G |
| 硬盘 | 1个 | 1 T |
| 操作系统 | 1个 | 银河麒麟4.0.1E |

### 4.2 应用介绍

验证系统包括四个功能, 词频统计、资源监控、数据恢复和动态部署与报警, 见表 2.

表 2 应用介绍

| 验证项目 | 说明 |
|---|---|
| 词频统计 | 卫星群分别运用单机和分布式对英文小说的词频进行计算, 通过所用时间的对比, 直观显示卫星群分布式计算的优势. |
| 资源监控 | 对卫星群进行分布式资源监控, 便于分析集群的运行状态并做故障的检测与排查. |
| 数据恢复 | 模拟卫星文件损坏后, 其他卫星对其的恢复过程. 展现卫星群分布式存储的安全性. |
| 动态部署与报警 | 通过动态部署体现卫星群分布式架构的灵活性. 节点在卫星群内标记为绿色, 表明活跃状态, 否则标记为红色, 表明宕机状态. |

### 4.3 界面迁移

验证系统的界面迁移如图 11 所示.

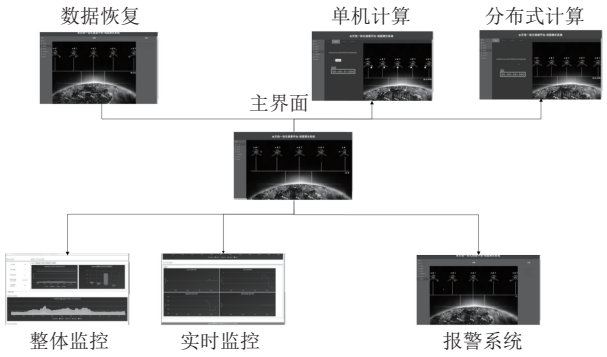


图 11 界面迁移图

## 5 实验结果

### 5.1 卫星群分布式资源监控结果

卫星群资源监控整体界面如图 12 所示, 该监控系统可对卫星群进行整体、实时、单点和不同节点的对比监控, 通过全方位监控卫星群, 保证其安全可靠的运行.

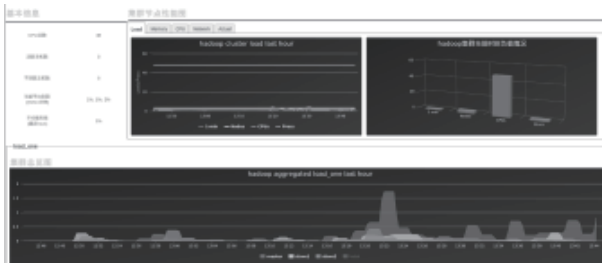

图 12 卫星群资源监控图

### 5.2 动态部署与报警结果

如图 13 所示, 选中卫星 2 与与地面三台设备后, 这四台机器构成一个星地集群, 其他卫星被移除集群, 并显示红色进行报警, 选中节点显示绿色表示正常运行.

### 5.3 卫星群分布式计算结果分析

以词频统计的测试程序为例, 展示卫星群分布式计算的优越性. 用动态部署所选中的星地四台节点, 对大小 350 Mb 的文本文件进行词频统计的分布式计算, 用时 72 s, 单机计算 242 s, 卫星群分布式计算增速三倍有余. 图 14 为卫星单机与分布式计算对比图.

卫星分布式计算性能与文件大小关联性见表 3.

卫星群分布式计算环境下, 当计算文件小于 125 M 时, 单机反而快于分布式计算, 这是是因为此时节点间的通信开销远大于计算开销. 当文件大于 125 M 之后, 分布式计算逐渐快于单机计算, 且在 300 MB 文件左右, 提高的速度趋于稳定, 约为单机性能的 3.3 倍. 这是合理的, 因为卫星群分布式计算需要占取网络和存储资源, 理论上不可能达到 4 倍的提升速度.

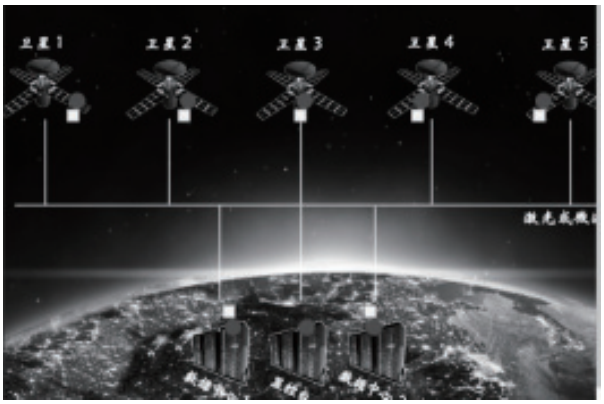

图 13 卫星动态部署与报警

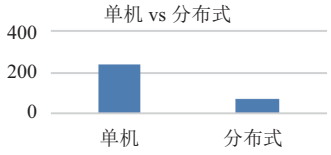


图 14 计算对比图

表 3 文件大小与性能关系表

| 文件大小(MB) | 单机(s) | 分布式(s) |
|---|---|---|
| 50 | 22 | 38 |
| 100 | 41 | 45 |
| 125 | 56 | 56 |
| 150 | 85 | 58 |
| 200 | 122 | 63 |
| 250 | 168 | 66 |
| 300 | 203 | 69 |
| 350 | 242 | 72 |
| 400 | 288 | 91 |
| 450 | 320 | 103 |
| 500 | 369 | 115 |
| 600 | 401 | 128 |

### 5.4 卫星群分布式存储结果

传统卫星单机存储技术, 如果数据损坏无法第一时间得知, 且需由地面上传数据. 而卫星群分布式存储技术, 始终保持数据的备份数为 3, 当某一节点丢失数据后, 最近的节点会迅速进行数据的自动恢复. 图 15 展示了将 ip 地址为 192.168.1.2 的卫星 2 的 blk_1073741835_1011 文件删除后, ip 地址为 192.168.1.5 的卫星 3 自动将丢失数据传送给卫星 2 的数据恢复过程.


图 15 数据恢复日志

## 6 结论

本文通过选取符合卫星条件限制的处理器架构和操作系统, 模拟搭建了卫星群的分布式计算环境, 运用现有的分布式技术, 实现了卫星群的动态部署、分布式计算、存储和资源监控功能, 并对卫星群资源监控功能进行了改进和扩展. 通过卫星群验证系统的实验, 证明了分布式技术在国产卫星平台上部署的可行性. 此外通过卫星群与单机卫星节点在计算时间和数据恢复上的对比, 证明了分布式技术在国产卫星平台性能上的优越性.

## 参考文献

1 王敬超, 于全. 基于分布式星群的空间信息网络体系架构与关键技术. 中兴通讯技术, 2016, 22(4): 9–13, 18.
2 彭东. 深度探索嵌入式操作系统. 北京: 机械工业出版社, 2015.
3 陆嘉恒. Hadoop 实战. 2 版. 北京: 机械工业出版社, 2012.
4 White T. Hadoop 权威指南. 3 版. 华东师范大学数据科学与工程学院译. 北京: 清华大学出版社, 2015.
5 Holmes A. Hadoop 硬实战. 梁李印, 宁青, 杨卓荦, 译. 北京: 电子工业出版社, 2015.
6 顾炯炯. 云计算架构技术与实践. 北京: 清华大学出版社, 2016.
7 Zandstra M. 深入 PHP. 人民邮电出版社, 2016.
8 尤海鹏. 基于 Ganglia 的数据中心监控平台设计[硕士学位论文]. 济南: 山东大学, 2014.
9 曹东航. 基于 Ganglia 的云平台监控的研究与实现[硕士学位论文]. 成都: 电子科技大学, 2016.