**LustrePerfMon**

用户手册

**Version 0.0.10| ref.notbd | Revision A0**



目录

[目录 3](#_Toc27930099)

[1 EXAScalerx性能监控系统简介 10](#_Toc27930101)

[1.1 名词解释 10](#_Toc27930102)

[1.2 Collectd插件 11](#_Toc27930103)

[2 安装要求 12](#_Toc27930104)

[2.1 部署服务器 12](#_Toc27930105)

[2.2 监控服务器 12](#_Toc27930106)

[2.3 监控代理节点 12](#_Toc27930107)

[2.4 对SFA的要求 12](#_Toc27930108)

[3 安装过程 13](#_Toc27930109)

[3.1 准备部署服务器 13](#_Toc27930110)

[3.2 准备监控服务器 13](#_Toc27930111)

[3.3 更新配置文件 13](#_Toc27930112)

[3.4 在集群上运行安装程序 16](#_Toc27930113)

[3.5 访问监控网络页面 17](#_Toc27930114)

[4 仪表盘 18](#_Toc27930115)

[4.1 集群状态仪表盘 18](#_Toc27930116)

[4.2 Lustre仪表盘 19](#_Toc27930117)

[4.3 Lustre MDS仪表盘 41](#_Toc27930118)

[4.4 Lustre OSS仪表盘 56](#_Toc27930119)

[4.5 服务器仪表盘 71](#_Toc27930120)

[4.6 SFA 物理磁盘仪表盘 76](#_Toc27930121)

[4.7 SFA虚拟磁盘表盘 79](#_Toc27930122)

[5 压力测试 82](#_Toc27930123)

[5.1 在代理节点安装插件 82](#_Toc27930124)

[5.2 在代理节点更新配置文件 82](#_Toc27930125)

[5.3 进行压力测试 83](#_Toc27930126)

[6 故障排查 86](#_Toc27930127)

图片列表

[图1: Grafana登陆界面 17](#_Toc27930128)

[图2：主仪表盘 18](#_Toc27930129)

[图3：集群状态仪表盘 19](#_Toc27930130)

[图4：Lustre仪表盘 19](#_Toc27930131)

[图5: Lustre系统总剩余容量面板 20](#_Toc27930132)

[图6: Lustre文件系统已使用总容量面板 20](#_Toc27930133)

[图7: Lustre文件系统每个OST剩余容量面板 21](#_Toc27930134)

[图8: 每个OST已使用容量面板 21](#_Toc27930135)

[图9: 用户已使用容量面板 22](#_Toc27930136)

[图10: 用户组已使用容量面板 22](#_Toc27930137)

[图11: 总的剩余索引节点数目面板 23](#_Toc27930138)

[图12: 总的已使用索引节点数目面板 23](#_Toc27930139)

[图13: 每个MDT剩余索引节点数目面板 24](#_Toc27930140)

[图14: 每个用户已使用索引节点数目面板 24](#_Toc27930141)

[图15: 每个用户组已使用索引节点数目面板 25](#_Toc27930142)

[图16: 每个MDT已使用索引节点数目面板 25](#_Toc27930143)

[图17: Lustre文件系统总的I/O吞吐量面板 26](#_Toc27930144)

[图18: 每个OST的I/O吞吐量面板 26](#_Toc27930145)

[图19: 每个OST写吞吐量面板 27](#_Toc27930146)

[图20: 每个OST读吞吐量面板 27](#_Toc27930147)

[图21: 总的元数据操作速率面板 28](#_Toc27930148)

[图22: 每个MDT的元数据操作速率面板 28](#_Toc27930149)

[图23: 每个客户端的元数据操作速率面板 29](#_Toc27930150)

[图24: 每种类型元数据操作速率面板 29](#_Toc27930151)

[图25: 不同块大小的写RPC速率面板 30](#_Toc27930152)

[图26: 写bulk RPC大小分布面板 30](#_Toc27930153)

[图27: 不同大小的读bulk RPC速率面板 31](#_Toc27930154)

[图28: 读bulk RPC大小分布面板 31](#_Toc27930155)

[图29: 写I/O不连续页面分布面板 32](#_Toc27930156)

[图30: 读I/O不连续页面分布面板 32](#_Toc27930157)

[图31: 写I/O不连续块分布面板 33](#_Toc27930158)

[图32: 读I/O不连续块分布面板 33](#_Toc27930159)

[图33: 每个写I/O碎片分布面板 34](#_Toc27930160)

[图34: 读I/O碎片分布面板 34](#_Toc27930161)

[图35: 已提交等待结束的磁盘写I/O分布面板 35](#_Toc27930162)

[图36: 已提交等待结束的磁盘读I/O分布面板 35](#_Toc27930163)

[图37: 写I/O时间分布面板 36](#_Toc27930164)

[图38: 读I/O时间分布面板 36](#_Toc27930165)

[图39: 磁盘写I/O大小分布面板 37](#_Toc27930166)

[图40: 磁盘读I/O大小分布面板 37](#_Toc27930167)

[图41: 每个客户端写吞吐量面板 38](#_Toc27930168)

[图42: 每个客户端读吞吐量面板 38](#_Toc27930169)

[图43: 每个作业I/O吞吐量面板 39](#_Toc27930170)

[图44: 每个作业写吞吐量面板 39](#_Toc27930171)

[图45: 每个作业读吞吐量面板 40](#_Toc27930172)

[图46: 每个作业元数据性能面板 40](#_Toc27930173)

[图47: Lustre MDS仪表盘 41](#_Toc27930174)

[图48：活跃请求数目面板 41](#_Toc27930175)

[图49：流入请求数目面板 42](#_Toc27930176)

[图50：请求等待时间面板 42](#_Toc27930177)

[图51：自适应超时值面板 43](#_Toc27930178)

[图52：可用请求缓冲区数目面板 43](#_Toc27930179)

[图53：LDLM ibits Enqueue请求处理时间面板 44](#_Toc27930180)

[图54：Getattr请求处理时间面板 45](#_Toc27930181)

[图55：Connect请求处理时间面板 45](#_Toc27930182)

[图56：Get-root请求处理时间面板 46](#_Toc27930183)

[图57：Statfs请求处理时间面板 46](#_Toc27930184)

[图58：Getxattr请求处理时间面板 47](#_Toc27930185)

[图59：Ping请求处理时间面板 47](#_Toc27930186)

[图60：活跃的Readpage请求数目面板 48](#_Toc27930187)

[图61：流入的Readpage请求数目面板 48](#_Toc27930188)

[图62：Readpage请求等待时间面板 49](#_Toc27930189)

[图63：Readpage自适应超时值面板 49](#_Toc27930190)

[图64：可用Readpage请求缓冲区数目面板 50](#_Toc27930191)

[图65：Close请求处理时间面板 50](#_Toc27930192)

[图66：Readpage请求处理时间面板 51](#_Toc27930193)

[图67：活跃的LDLM Canceld请求数目面板 51](#_Toc27930194)

[图68：流入的LDLM Canceld请求数目面板 52](#_Toc27930195)

[图69：LDLM Canceld请求等待时间面板 52](#_Toc27930196)

[图70：LDLM Canceld服务的自适应超时值面板 53](#_Toc27930197)

[图71：可用LDLM Canceld请求缓冲区数目面板 53](#_Toc27930198)

[图72：活跃的LDLM Callback请求数目面板 54](#_Toc27930199)

[图73：流入的LDLM Callback请求数目面板 54](#_Toc27930200)

[图74：LDLM Callback请求等待时间面板 55](#_Toc27930201)

[图75：LDLM Callback自适应超时值面板 55](#_Toc27930202)

[图76：可用LDLM Callback请求缓冲区数目面板 56](#_Toc27930203)

[图77: Lustre OSS仪表盘 56](#_Toc27930204)

[图78：I/O带宽面板 57](#_Toc27930205)

[图79: 活跃请求数目面板 57](#_Toc27930206)

[图80: 流入请求数目面板 58](#_Toc27930207)

[图81: 请求等待时间面板 58](#_Toc27930208)

[图82: 自适应超时值面板 59](#_Toc27930209)

[图83: 可用请求缓冲区数目面板 59](#_Toc27930210)

[图84: 活跃的I/O请求数目面板 60](#_Toc27930211)

[图85: 流入的I/O请求数目面板 60](#_Toc27930212)

[图86: I/O请求等待时间面板 61](#_Toc27930213)

[图87: I/O服务的自适应超时值面板 61](#_Toc27930214)

[图88: 可用I/O请求缓冲区数目面板 62](#_Toc27930215)

[图89: Punch请求处理时间面板 62](#_Toc27930216)

[图90: 读请求处理时间面板 63](#_Toc27930217)

[图91: 写请求处理时间面板 63](#_Toc27930218)

[图92: 活跃的创建请求数目面板 64](#_Toc27930219)

[图93: 流入的创建请求数目面板 64](#_Toc27930220)

[图94: 创建请求等待时间面板 65](#_Toc27930221)

[图95: 创建服务的自适应超时值面板 65](#_Toc27930222)

[图96: 可用创建请求缓冲区数目面板 66](#_Toc27930223)

[图97: 活跃的LDLM Canceld请求数目面板 66](#_Toc27930224)

[图98: 流入的LDLM Canceld请求数目面板 67](#_Toc27930225)

[图99: LDLM Canceld请求等待时间面板 67](#_Toc27930226)

[图100: LDLM Canceld服务自适应超时值面板 68](#_Toc27930227)

[图101: 可用的LDLM Canceld请求缓冲区数目面板 68](#_Toc27930228)

[图102: 活跃的LDLM Callback请求数目面板 69](#_Toc27930229)

[图103: 流入的LDLM Callback请求数目面板 69](#_Toc27930230)

[图104: LDLM Callback请求等待时间面板 70](#_Toc27930231)

[图105: LDLM Callback服务自适应超时值面板 70](#_Toc27930232)

[图106: 可用的LDLM Callback请求缓冲区数目面板 71](#_Toc27930233)

[图107：服务器仪表盘 71](#_Toc27930234)

[图108：CPU 使用率面板 72](#_Toc27930235)

[图109：内存使用率面板 72](#_Toc27930236)

[图110：磁盘写速率面板 73](#_Toc27930237)

[图111：磁盘读速率面板 73](#_Toc27930238)

[图112：根目录磁盘使用率 74](#_Toc27930239)

[图113：负载面板 74](#_Toc27930240)

[图114：启动时间面板 75](#_Toc27930241)

[图115：用户数面板 75](#_Toc27930242)

[图116：温度面板 75](#_Toc27930243)

[图117: SFA物理磁盘仪表盘 76](#_Toc27930244)

[图118：I/O 吞吐率面板 76](#_Toc27930245)

[图119：IOPS面板 77](#_Toc27930246)

[图120：I/O 大小面板 77](#_Toc27930247)

[图121：写数据吞吐率面板 78](#_Toc27930248)

[图122：写数据的I/O大小样品采样数目面板 78](#_Toc27930249)

[图123：数据写入延迟样品采样数目面板 78](#_Toc27930250)

[图124：SFA虚拟磁盘表盘 79](#_Toc27930251)

[图125：I/O吞吐率面板 79](#_Toc27930252)

[图126：IOPS面板 80](#_Toc27930253)

[图127：虚拟磁盘每次I/O的字节数面板 80](#_Toc27930254)

[图128：虚拟磁盘的写入性能面板 80](#_Toc27930255)

[图129：数据写入的I/O大小样品采集数目面板 81](#_Toc27930256)

[图130：数据写入的延迟样品采集数目面板 81](#_Toc27930257)

[图131：Influxdb Query Time 变化曲线图 85](#_Toc27930258)

[图132：每个任务读吞吐压测面板 85](#_Toc27930259)

# EXAScalerx性能监控系统简介

LustrePerfMon 是一款监控系统，通过采集Exascaler的系统状态信息以达到对其进行性能监控及分析的目的。LustrePerfMon基于多种开源软件的监控系统，同时还提供了一些外部插件以作功能扩展。

LustrePerfMon的主要组件之一是 **collectd**。**collectd**是一个运行在监控对象上的守护进程，完成了系统性能的相关统计信息定期收集，并将这些数据以多种不同的机制进行存储。LustrePerfMon是基于开源的**collectd**而研发，同时也包含了许多其他插件，如Filedata、Ganglia、Nagios、Stress、Zabbix等等。

## 名词解释

|  |  |
| --- | --- |
| **LustrePerfMon** | Exascaler性能监控系统的缩写。 |
| **SFA** | Storage Fusion Architecture的缩写。Storage Fusion Architecture为平衡的高性能存储提供基础。通过使用并行存储处理技术，SFA提供了强大的IOPS和吞吐率。 |
| **EXAScaler** | Exascaler是基于Lustre的存储解决方案，它旨在解决极端数据密集型环境中最严苛的存储及数据管理问题。 |
| **部署服务器** | LustrePerfMon监控系统的安装进程将在此服务器上被触发。 |
| **监控服务器** | LustrePerfMon监控系统的数据库(Influxdb) 及网络服务 (Grafana) 将在此服务器上运行。 |
| **代理节点** | LustrePerfMon监控系统将从代理节点上收集各类指标数据，例如CPU, 内存、Lustre、SFA 存储等相关信息。守护进程collectd 将在每个客户端后台运行。 |
| **IME** | 无限内存引擎(Infinite Memory Engine) 是一款基于闪存的高速缓存系统，它简化了应用程序I/O路径，消除了系统瓶颈。 |
| **Lustre** | Lustre文件系统是一种开源的并行文件系统，它满足了许多高性能计算环境的海量数据存储需求。 |
| **OST** | Lustre的对象存储目标（Object Storage Target）是用来存储文件数据对象的存储目标。 |
| **OSS** | Lustre的对象存储服务器（Object Storage Server）是用来管理对象存储目标的服务器。 |
| **MDT** | Lustre的元数据目标（Metadata Target）是用来存储文件元数据的存储目标。 |
| **MDS** | Lustre的元数据服务器（Metadata Server）是用来为文件系统提供元数据服务，管理一个或多个元数据目标的服务器 |

## Collectd插件

为支持更多不同功能，我们还提供了一些附加的Collectd插件。

* **Filedata插件**：**Filedata**插件可通过读取及解析一组文件进行数据收集。用户需要在一个xml格式文件中对读取哪些文件、如何解析这些文件等进行定义。**Filedata**插件最常见的用途是通过正在运行的Lustre系统的/proc接口收集指标。
* **Ganglia插件**：Ganglia插件将collectd进程收集的指标信息发送给Ganglia服务器。
* **IME插件**：**IME**插件通过 **IME**收集性能信息。**IME**插件和**Filedata**插件共享相似的定义文件格式和配置格式。
* **SSH插件**：SSH插件能够通过使用SSH连接在远程主机上运行命令来收集来自SFA存储的各项指标。与**IME**插件一样，它的定义文件格式和配置格式和Filedata插件类似。
* **Stress插件：Stress**插件可以从**collectd**客户端向服务器推送大量数据，来对监控系统性能进行高强度基准测试。
* **Stress2插件：Stress** 插件改进版，可供灵活配置数据格式，仿真各种插件从客户端向服务器推送大量数据。
* **Zabbix插件**：Zabbix插件将指标数据从**collectd**发送至**Zabbix**系统。

# 安装要求

## 部署服务器

* **操作系统版本：**CentOS7/RHEL7。
* **硬盘空闲空间：**>500MB。所有安装日志将被保存于部署服务器的*/var/log/esmon\_install*目录下，这需要占用一定空间。
* **网络：**部署服务器须能对监控服务器和被监控客户端发起无密码提示的SSH连接。
* **LustrePerfMon ISO镜像：**部署服务器上需有可用的LustrePerfMon ISO 镜像。
* **时钟和时区：**节点的时钟和时区需要和其他节点一致。如果时钟或时区不一致，可能会导致数据收集或显示异常。

## 监控服务器

* **操作系统版本：** CentOS7/RHEL7。
* **硬盘空闲空间：**> 5GB。监控服务器上会运行Influxdb。如果Influxdb需容纳更多数据，则须预留更大空间。
* **网络：**监控服务器须运行SSHD。部署服务器必须可以通过无密码提示的SSH与监控服务器连接。
* **时钟和时区：**节点的时钟和时区需要和其他节点一致。

## 监控代理节点

* **操作系统版本：**CentOS7/RHEL7或CentOS6/RHEL6。
* **硬盘空闲空间：**> 200MB。必要的RPMs将被保存于*/var/log/esmon\_install*目录下，这将占用一定空间。
* **网络：**监控代理节点必须运行SSHD。部署服务器必须可以通过无密码提示的SSH与监控代理节点连接。
* **EXASCALER版本：**EXAScaler 2.x、EXAScaler 3.x或EXAScaler 4.x。
* **时钟和时区：**节点的时钟和时区需要和其他节点一致。

## 对SFA的要求

* **固件发行版：**3.x或11.x

# 安装过程

## 准备部署服务器

1. 将 LustrePerfMon ISO 镜像文件拷贝至部署服务器上，如*/ISOs/esmon.iso*。
2. 在部署服务器上挂载 LustrePerfMon ISO 镜像：

mount -o loop /ISOs/esmon.iso /media

1. 在部署服务器上，如果存在旧的LustrePerfMon配置文件，备份之：

cp /etc/esmon\_install.conf /etc/esmon\_install.conf\_backup

1. 在部署服务器上，如果已安装旧的LustrePerfMon RPM包，卸载之：

rpm –e esmon

1. 在部署节点上安装 LustrePerfMon RPM：

rpm -ivh /media/RPMS/rhel7/esmon\*.rpm

## 准备监控服务器

如果监控服务器的防火墙打开，那么需要将防火墙的3000、4242、8086、8088、25826等端口打开，否则LustrePerfMon的安装或者使用会出现问题。其中3000端口为Grafana的服务端口，4242、8086、8088、25826等端口为Influxdb、Grafana、Collectd之间进行数据传输和管理的端口。

## 更新配置文件

在安装LustrePerfMon的RPM之后，更新配置文件/etc/esmon\_install.conf。该文件包含了安装所需的所有必要信息。定义如下参数：

* 在**agents**部分，定义了需要安装和配置LustrePerfMon代理的所有主机的信息：
  + **enable\_disk** — 定义了是否启用磁盘指标收集功能，默认值为**false**。
  + **host\_id** — 该主机的唯一标识。两个不同的主机不能拥有相同的**host\_id**。
  + **ime** — 定义了是否启用**IME**中指标收集功能，默认值为**false**。
  + **infiniband** — 定义了是否启用**Infiniband**中指标收集功能，默认值为**false**。
  + **lustre\_mds** — 定义了是否启用 **Lustre MDS**上的数据收集功能，默认值为**true**。
  + **lustre\_oss** — 定义了是否启用 **Lustre OSS**上的数据收集功能，默认值为**true**。
  + **sfas** — 这个列表包含了该代理上一个或多个SFA的相关信息。
    - **controller0\_hos** — 定义了该SFA控制器0的主机名或IP。
    - **controller1\_host** — 定义了该SFA控制器1的主机名或IP。
    - **name** — 定义了该SFA的唯一名字。该名字用作该SFA的“fqdn”标签值。两个SFA不应拥有相同的名字。
* **agents\_reinstall** — 定义了是否重装这些代理节点，默认值**true**。
* **collect\_interval** — 定义了收集数据点的时间间隔（秒），默认值**60**。
* **continuous\_query\_interval** — 定义了连续查询（Continuous Query）的时间间隔。LustrePerfMon使用Influxdb的连续查询来聚合数据。要得到连续查询的间隔时间秒数，请将该参数值乘以**collect\_interval**的数值。如果本参数值为1，那么连续查询的间隔时间秒数即为**collect\_interval**秒。为了降低采样频率，减少性能压力，该值通常设置为大于1的数值，默认值为**4**。
* **iso\_path** — LustrePerfMon ISO镜像的保存路径，默认值为**/root/esmon.iso**。
* **lustre\_default\_version** — 定义了默认的Lustre版本。如果在代理节点上安装的Lustre RPM包未匹配已支持版本，则使用默认Lustre版本。目前支持的参数值有：es2，es3，es4和error。如果配置的值为error, 当代理节点使用了一个不支持的Lustre版本时，会出现报错。默认值为es3。
* **lustre\_exp\_ost** —­­­ 定义了是否启用从代理节点的Lustre OST上收集exp\_ost\_stats\_[read|write]指标功能。如果Lustre文件系统有太多客户端，为避免性能问题，该选项应关闭。默认值为**false**。
* **lustre\_exp\_mdt** —­­­ 定义了是否启用从代理节点的Lustre MDT上收集exp\_md\_stats\_\*指标功能。如果Lustre文件系统有太多客户端，为避免性能问题，该选项应关闭。默认值为**false**。
* 在**server**部分，定义了监控服务器的信息：
  + **drop\_database** — 定义了是否丢弃Influxdb中已有的LustrePerfMon数据库，默认值为**false**。

**重要:** 只有当不需要保留Influxdb中数据和元数据时才可以启用**drop\_database。**

* + **erase\_influxdb** — 定义了是否完全清除所有Influxdb中的数据和元数据，默认值为**False**。当Influxdb完全毁坏时，打开该选项可以清除数据库恢复Influxdb的运行。

**重要:** 只有当不需要保留Influxdb中数据和元数据时才可以启用**erase\_influxdb**。在打开该选项之前，请仔细检查influxdb\_path设置正确。

* + **host\_id** — LustrePerfMon服务器主机的唯一标识。
  + **influxdb\_path** — 定义了LustrePerfMon服务器主机上的Influxdb目录路径，默认值为**/esmon/influxdb**。

**重要: 请勿在该目录下放置其他文件或目录，因为在erase\_influxdb打开的情况下，该目录下的所有子目录和文件将被清除。**

* + **reinstall** — 定义了是否重新安装LustrePerfMon服务器，默认值为**true**。
* 在**ssh\_hosts**部分，详细定义了用如何从部署服务器上用SSH连接登录监控服务器、监控代理节点的相关信息：
  + **host\_id** — 该主机的唯一标识。两个不同的主机不能拥有相同的**host\_id**。
  + **hostname** — 进行SSH连接时的主机名或IP。“ssh”命令中会使用该主机名或IP来登入主机。如果该主机用作LustrePerfMon服务器，那么该主机名或IP将作为服务节点配置到代理节点上的write\_tsdb插件里。
  + **ssh\_identity\_file** — 进行SSH连接时的使用的SSH密钥文件。如果默认的SSH密钥文件可用，该参数可设置为**None**。默认值为**None**。
  + **local\_host** —定义了该主机是否为本地节点。本地节点将不需要SSH连接。默认值为**false**。

注意: 对同一个主机，**host\_id**和**hostname**可能不同，因为可以用多种方式连接至同一个主机。

下面是/etc/esmon\_install.conf的一个例子

**例子:**

agents:

- enable\_disk: false

host\_id: Agent1

ime: false

infiniband: false

lustre\_mds: true

lustre\_oss: true

sfas:

- controller0\_host: 10.0.0.1

controller1\_host: 10.0.0.2

name: SFA1

- controller0\_host: 10.0.0.3

controller1\_host: 10.0.0.4

name: SFA2

- host\_id: Agent2

sfas: []

agents\_reinstall: true

collect\_interval: 60

continuous\_query\_interval: 4

iso\_path: /root/esmon.iso

lustre\_default\_version: es3

lustre\_exp\_mdt: false

lustre\_exp\_ost: false

server:

drop\_database: false

erase\_influxdb: false

host\_id: Server

influxdb\_path: /esmon/influxdb

reinstall: true

ssh\_hosts:

- host\_id: Agent1

hostname: Agent1

local\_host: false

ssh\_identity\_file: None

- host\_id: Agent2

hostname: Agent2

- host\_id: Server

hostname: Server

## 在集群上运行安装程序

在部署服务器上的*/etc/esmon\_install.conf*更新完成后，运行以下命令在集群上启动安装程序：

esmon\_install

所有可用于调试的相关日志将被保存在*/var/log/esmon\_install*目录下。

esmon\_install命令除了可以用来在新系统上安装LustrePerfMon外，还可以用来更新已有系统。为此，在安装完成LustrePerfMon之后，应备份*/etc/esmon\_install.conf*文件。

**重要:** 在升级已有LustrePerfMon系统时，**erase\_influxdb**和**drop\_database**应当关闭，除非Influxdb 中的数据和元数据不需要保留。

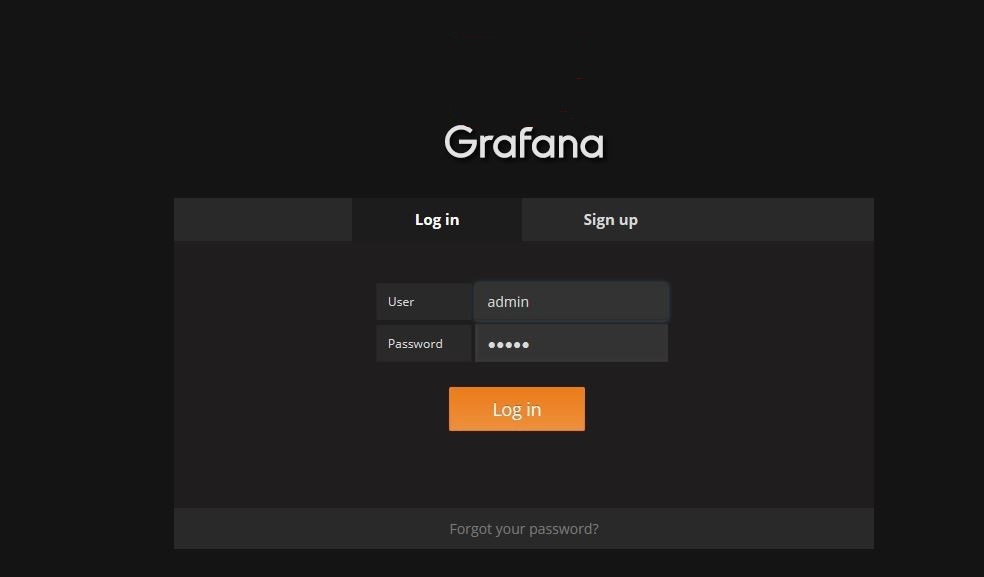
当安装或升级时，**esmon\_install**将会清除并安装所有默认的LustrePerfMon的Grafana仪表盘。除了默认的LustrePerfMon的Grafana仪表盘外，**esmon\_install**不会改变任何其他的已有Grafana仪表盘。

**重要:** 在升级已有LustrePerfMon系统之前，应通过Grafana网页界面备份所有经定制的LustrePerfMon默认仪表盘，将此复制为其他的名字，否则这些定制修改将会被升级过程所覆盖。

## 访问监控网络页面

Grafana服务将自动在监控服务器启动。默认HTTP 端口为3000。通过访问该端口可跳转至登录页面 (图1)，默认用户名密码皆为*admin*。

图1: Grafana登陆界面

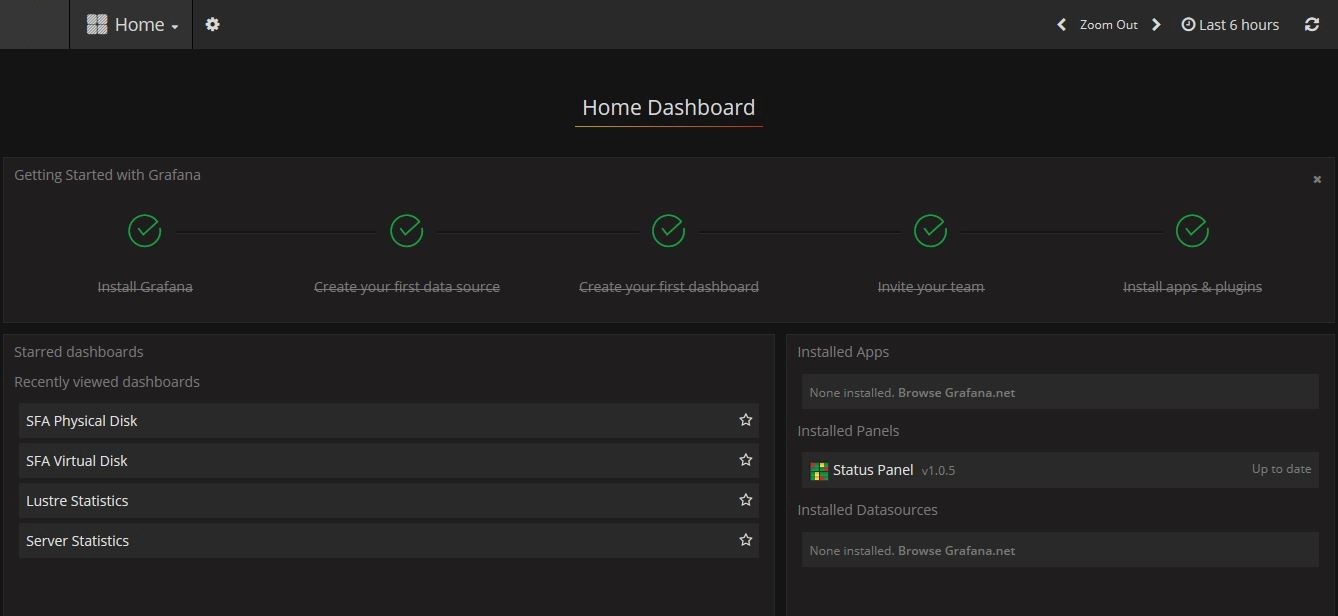


**重要:** 访问监控页面的节点的时钟和时区应与服务节点保持一致，否则数据的显示将有可能不正常。

# 仪表盘

在主仪表盘上（图2），可通过选择不同的仪表盘来浏览由ESMON收集的各种数据。

图2：主仪表盘

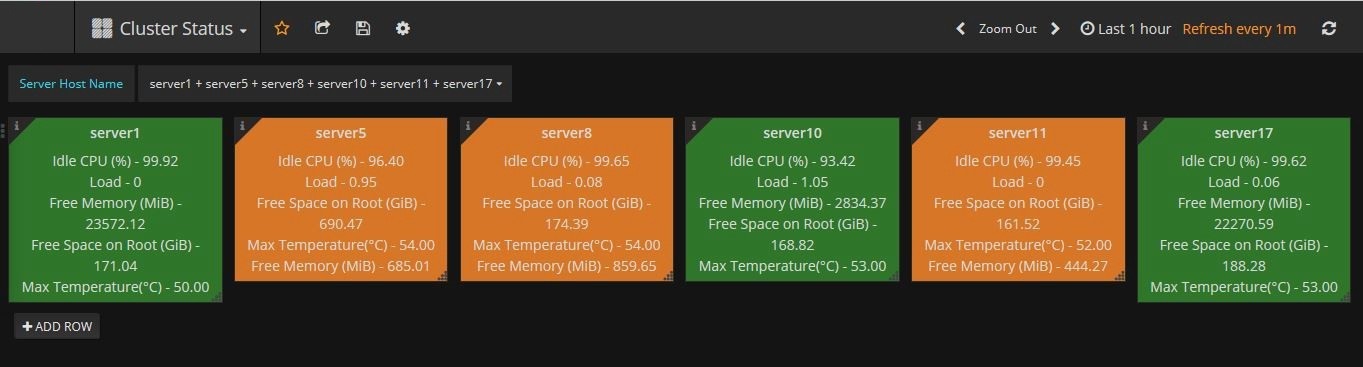


## 集群状态仪表盘

集群状态仪表盘（图3）显示了集群中服务器的状态信息概要。其中，面板的背景颜色与服务器的运行状态相关

* 绿色：服务器正常运行。
* 黄色：警告信息，说明发生一个或多个下列情况：
  + 空闲 CPU 不足20%；
  + 负载数量高于5；
  + 空闲内存不足 1000 MiB；
  + 根目录空闲空间不足10 GiB。
* 红色：严重警告信息，说明发生一个或多个下列情况：
  + 空闲 CPU 不足5%；
  + 负载数量高于10；
  + 根目录空闲空间不足1 GiB；
  + 空闲内存不足 100 MiB。

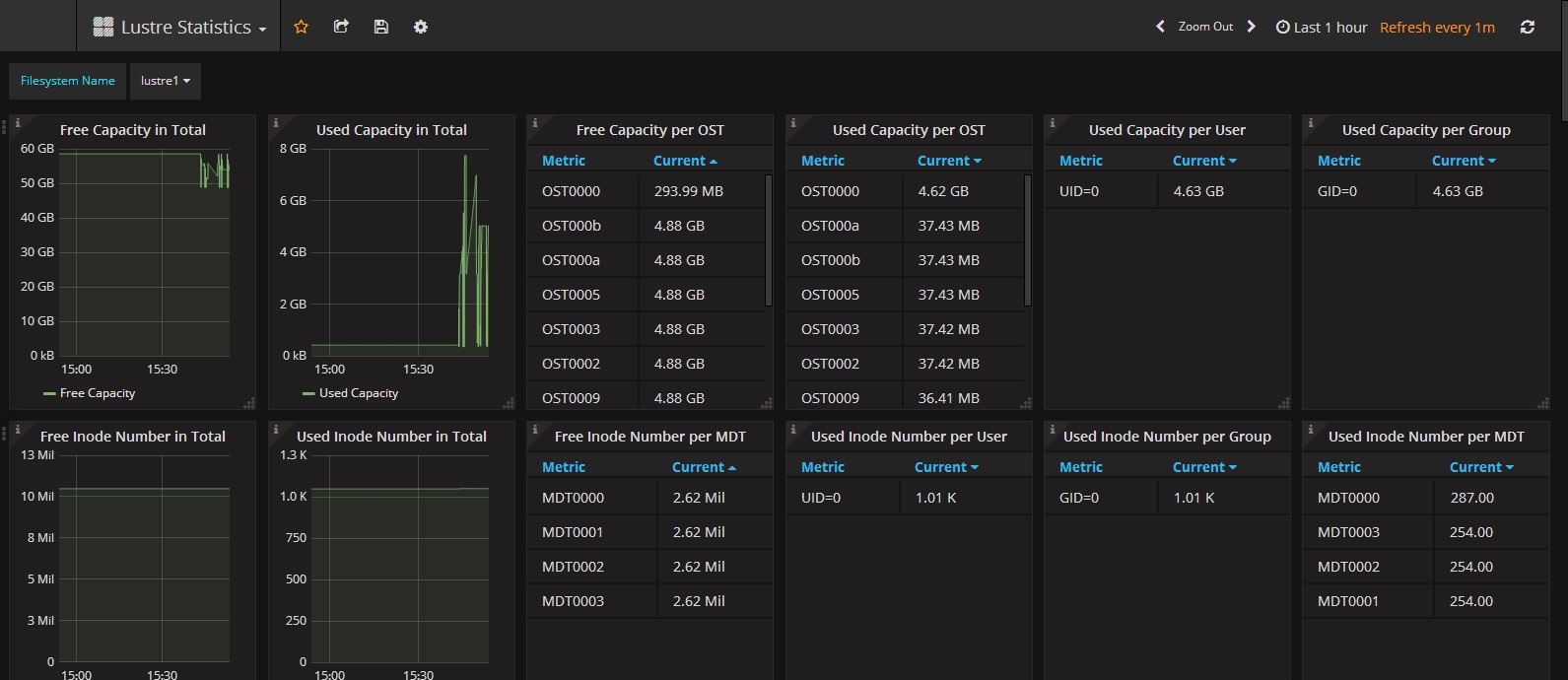
图3：集群状态仪表盘



## Lustre仪表盘

Lustre仪表盘（图4）显示了 Lustre 文件系统统计数据。

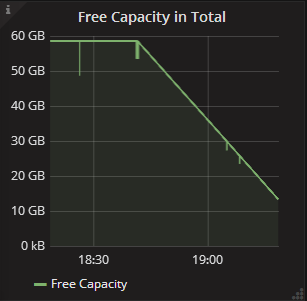
图4：Lustre仪表盘



以下是一些主要指标的面板视图：

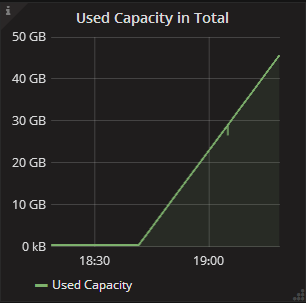
* **系统总剩余容量**面板(图5) 显示了Lustre文件系统剩余容量随时间的变化。从大约18:40开始运行测试用例“dd if=/dev/zero of=/mnt/lustre/file bs=1M”，图中显示出剩余容量正在以大约20MB/s的速度不断消耗。

图5: Lustre系统总剩余容量面板



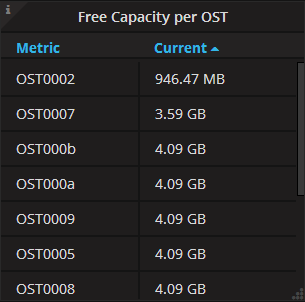
* **系统已使用总容量**面板(图6) 显示了Lustre文件系统已使用总容量随时间的变化。该图的测试用例为“dd if=/dev/zero of=/mnt/lustre/file bs=1M”，从图可以看出已使用容量从18:40开始以大约20MB/s的速度递增。

图6: Lustre文件系统已使用总容量面板



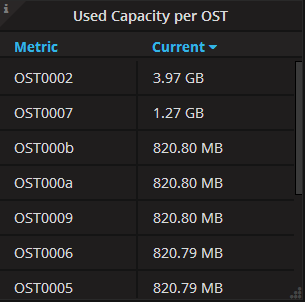
**每个OST剩余容量**面板(图7) 显示了Lustre文件系统每个OST剩余容量大小。如图所示，OST0002剩余容量为946.47MB， OST0007剩余容量为3.59GB，其他OST剩余容量都为4.09GB。点击**Current**，可以根据当前容量进行从小到大（或从大到小）排序。

图7: Lustre文件系统每个OST剩余容量面板



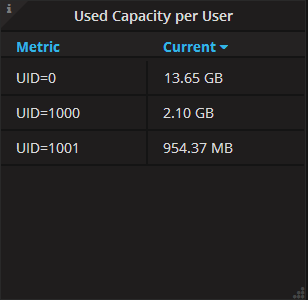
**每个OST已使用容量**面板(图8) 显示了Lustre文件系统每个OST已使用的容量大小。如图所示，OST0002已使用容量为3.97GB， OST0007已使用容量为1.27GB，其余OST已使用容量都为820.8MB。点击Current，可以根据当前已使用容量进行从小到大（或从大到小）排序。

图8: 每个OST已使用容量面板



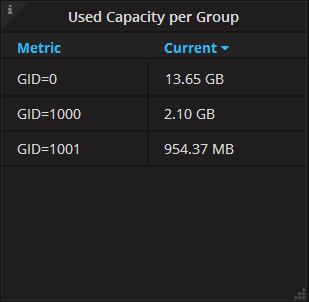
用户已使用容量面板(图9) 显示了Lustre文件系统每个用户已使用的容量大小。如图所示，UID为0的用户当前使用的容量为13.65GB；UID为1000的用户当前使用的容量为2.10GB；UID为1001的用户当前使用的容量为954.37MB。

图9: 用户已使用容量面板



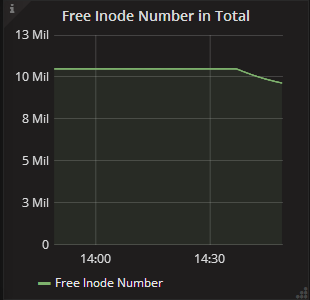
用户组已使用容量面板(图10) 显示了Lustre文件系统每个用户组已使用的容量大小。如图所示，GID为0的用户组当前使用的容量为13.65GB；GID为1000的用户组当前使用的容量为2.10GB；GID为1001的用户组当前使用的容量为954.37MB。

图10: 用户组已使用容量面板



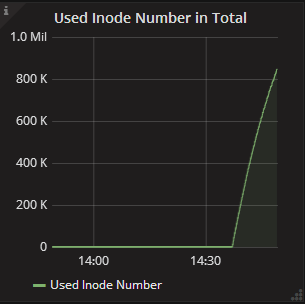
**总的剩余索引节点数目**面板(图11) 显示了Lustre文件系统总的剩余索引节点数目随时间的变化。该图所使用的测试用例为“mdtest–C –n 900000 –d /mnt/lustre/mdtest/”，从图可以看出从大约14:35开始运行测试用例后，剩余的索引节点数目以大约1100Ops的速度减少。

图11: 总的剩余索引节点数目面板



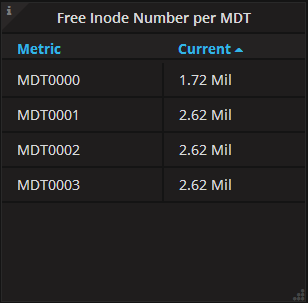
**总的已使用索引节点数目**面板(图12) 显示了Lustre文件系统总的已使用索引节点数目随时间的变化。该图所使用的测试用例为“mdtest–C –n 900000 –d /mnt/lustre/mdtest/”，从图可以看出从大约14:35开始运行测试用例后，已使用的索引节点数目以大约1100Ops的速度增加。

图12: 总的已使用索引节点数目面板



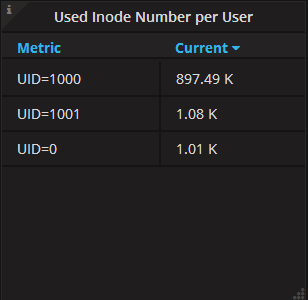
**每个MDT剩余索引节点数目**面板(图13) 显示了Lustre文件系统每个MDT剩余索引节点数目。如图所示，MDT0000剩余索引节点的数目为1.72Mil；系统其他各个MDT空闲的索引节点数目为2.62 Mil。点击Current，可以根据当前索引节点数目进行从小到大（或从大到小）排序。

图13: 每个MDT剩余索引节点数目面板



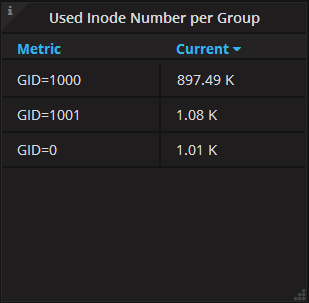
**每个用户已使用索引节点数目**面板(图14) 显示了Lustre文件系统每个用户已使用索引节点数目。如图所示，UID为1000的用户已使用的索引节点数目为897.49K； UID为1001的用户已使用的索引节点数目为1.08K；UID为0的用户已使用的索引节点数目为1.01K。点击Current，可以根据当前索引节点数目进行从小到大（或从大到小）排序。

图14: 每个用户已使用索引节点数目面板



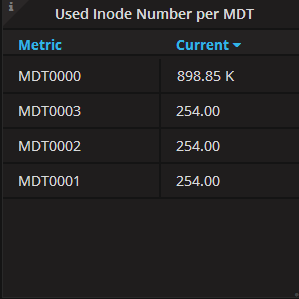
**每个用户组已使用的索引节点数目**面板(图15) 显示了Lustre文件系统每个用户组已使用的索引节点数目。如图所示，GID为1000的用户组已使用的索引节点数目为897.49K； GID为1001的用户组已使用的索引节点数目为1.08K；GID为0的用户组已使用的索引节点数目为1.01K。点击Current，可以根据当前索引节点数目进行从小到大（或从大到小）排序。

图15: 每个用户组已使用索引节点数目面板



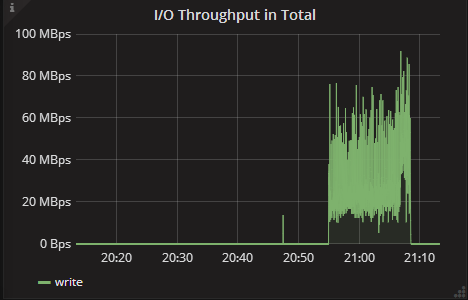
每个MDT已使用的索引节点数目面板(图16) 显示了Lustre文件系统每个MDT已使用的索引节点数目。如图所示，MDT0000使用索引节点数目为898.85K，MDT0001为254。

图16: 每个MDT已使用索引节点数目面板



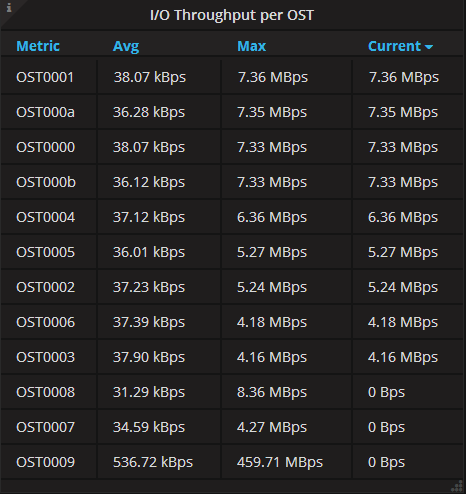
**总的I/O吞吐量**面板(图17) 显示了Lustre文件系统总的I/O吞吐量随时间的变化。

图17: Lustre文件系统总的I/O吞吐量面板



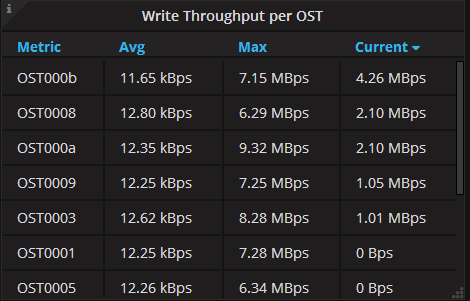
**每个OST的I/O吞吐量**面板(图18) 显示了Lustre文件系统每个OST的I/O吞吐量信息，包括平均值、最大值和当前值。

图18: 每个OST的I/O吞吐量面板



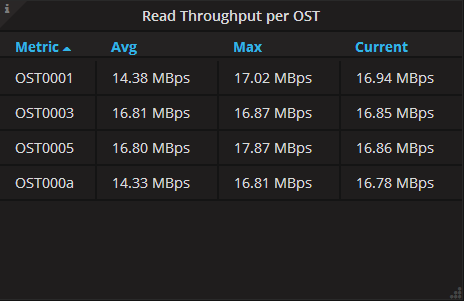
**每个OST的写吞吐量**面板(图19) 显示了Lustre文件系统每个OST的写吞吐量信息，包括平均值、最大值和当前值。

图19: 每个OST写吞吐量面板



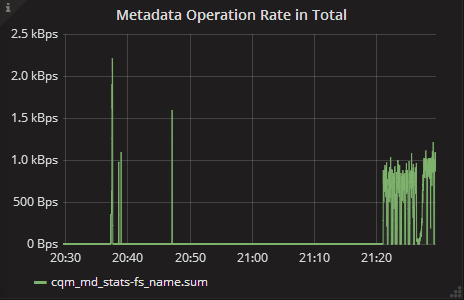
**每个OST的读吞吐量**面板(图20) 显示了Lustre文件系统每个OST的读吞吐量信息，包括平均值、最大值和当前值。

图20: 每个OST读吞吐量面板



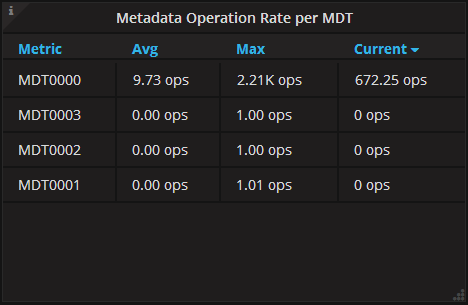
**总的元数据操作速率**面板(图21) 显示了Lustre文件系统总的元数据操作速率随时间的变化，其单位为Ops，即 Operation Per Second。

图21: 总的元数据操作速率面板



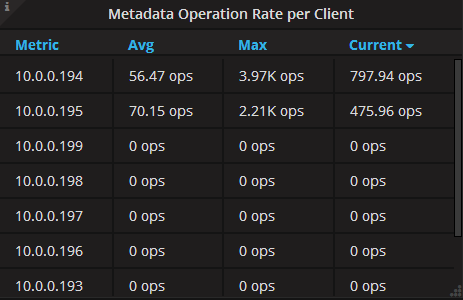
**每个MDT的元数据操作速率**面板(图22) 显示了Lustre文件系统每个MDT的元数据操作速率信息，其单位为Ops， Operation Per Second。其信息包括平均值、最大值和当前值。

图22: 每个MDT的元数据操作速率面板



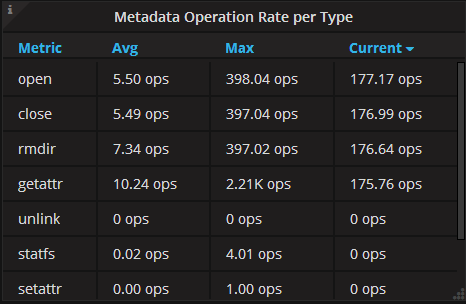
**每个客户端的元数据操作速率**面板(图23) 显示了Lustre文件系统每个客户端（client）的元数据操作速率信息，其单位为Ops，即 Operation Per Second。其信息包括平均值、最大值和当前值。

图23: 每个客户端的元数据操作速率面板



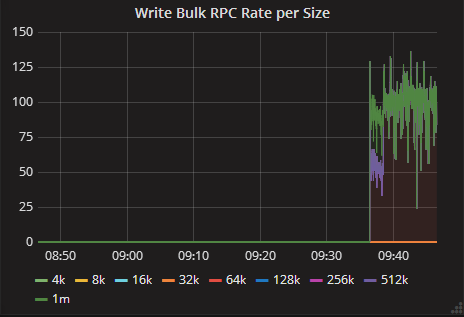
**每种类型的元数据操作速率**面板(图24) 显示了Lustre文件系统每种类型的元数据操作速率信息，其单位为Ops即Operation Per Second。其信息包括平均值、最大值和当前值。当前测试用例为删除一个目录下的所有文件的元数据操作速率统计。

图24: 每种类型元数据操作速率面板



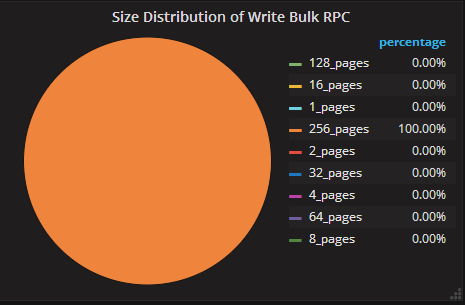
**不同块大小的写RPC速率**面板(图25) 显示了Lustre文件系统不同块大小的写RPC速率随时间的变化。Lustre文件系统统计了不同bulk RPC大小信息，从4K到16M，下图显示了不同bulk 大小的写RPC的速率信息。图中所示的测试用例为：在两个客户端分别运行“dd if=/dev/zero of=/mnt/lustre/test1 bs=1M oflag=direct”和“dd if=/dev/zero of=/mnt/lustre/test2 bs=64k oflag=direct”测试，收集到的信息。

图25: 不同块大小的写RPC速率面板



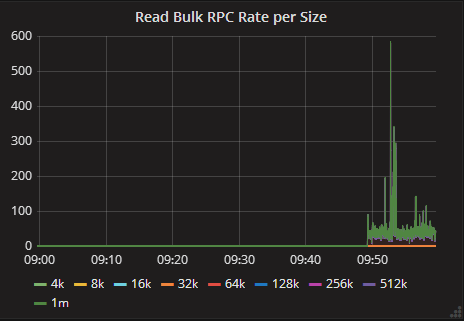
**写bulk RPC大小分布**面板(图26) 显示了Lustre文件系统不同bulk大小的写RPC数目的比例信息。如图所示，大小为256页的写bulk RPC数目占总的写RPC数目的百分比为100%。

图26: 写bulk RPC大小分布面板



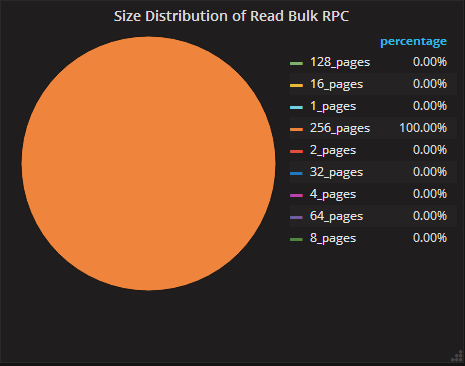
**不同大小的读bulk RPC速率**面板(图27) 显示了Lustre文件系统不同bulk大小的读RPC速率随时间的变化。Lustre文件系统统计了不同bulk RPC大小信息，从4K到16M（Lustre中最大RPC bulk I/O大小为16MB）。下图显示了不同bulk 大小的读RPC的速率信息。图中所示的测试用例为：在两个客户端分别运行“dd if=/mnt/lustre/test1 of=/dev/zero bs=1M iflag=direct”和“dd if=/mnt/lustre/test2 of=/dev/zero bs=64k iflag=direct”测试，收集到的信息。

图27: 不同大小的读bulk RPC速率面板



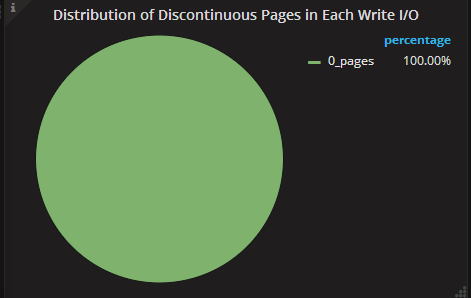
**读bulk RPC大小分布**面板(图28) 显示了Lustre文件系统不同bulk大小的读RPC数目比例信息。如图所示，该图正在进行的测试用例为“dd if=/mnt/lustre/file of=/dev/zero bs=1M”，大小为256页的读bulk RPC数目占总的RPC数目的百分比为100%。

图28: 读bulk RPC大小分布面板



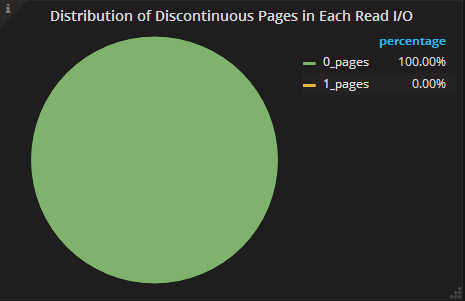
在Lustre 一次读写I/O中，如果写或者读的下一个页面与上一个读写页面的结尾（也就是下一个偏移量）不连续的话，那么就认为这个页面是不连续页面。在一次I/O RPC中可能有多个不连续页面。不连续的页面越少，底层的磁盘系统就能获取更好的性能。**写I/O不连续页面分布**面板(图29) 显示了Lustre文件系统写I/O不连续页面数目比例信息。如图所示，不连续页面为“0\_pages”数目占的百分比为100%，说明所有的页面都是连续的。

图29: 写I/O不连续页面分布面板



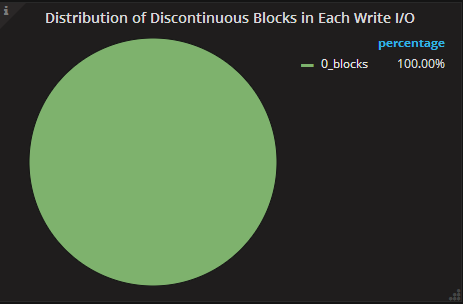
**读I/O不连续页面分布面板**(图30) 显示了Lustre文件系统读I/O不连续页面数目比例信息。如图所示，不连续页面为“0\_pages”数目占的百分比为100%，说明几乎所有的读页面都是连续的。

图30: 读I/O不连续页面分布面板



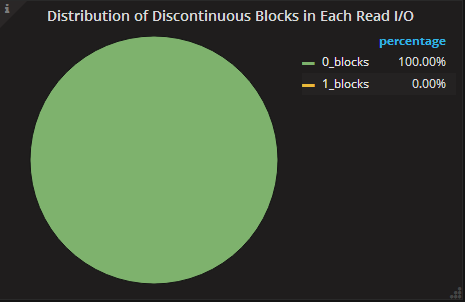
**写I/O不连续块分布面板**(图31) 显示了Lustre文件系统写I/O不连续块的数目比例信息。在Lustre 一次读写I/O中，不连续块的含义与不连续页的含义类似。一个块中含有的页面数量由底层文件系统（Ldiskfs）决定。I/O中存在不连续块，则意味着这里一定存在不连续页；但反之不一定成立。如图所示，写I/O不连续块为“0\_blocks”数目占的块I/O百分比为100%，说明几乎所有的写I/O都都是连续的。

图31: 写I/O不连续块分布面板



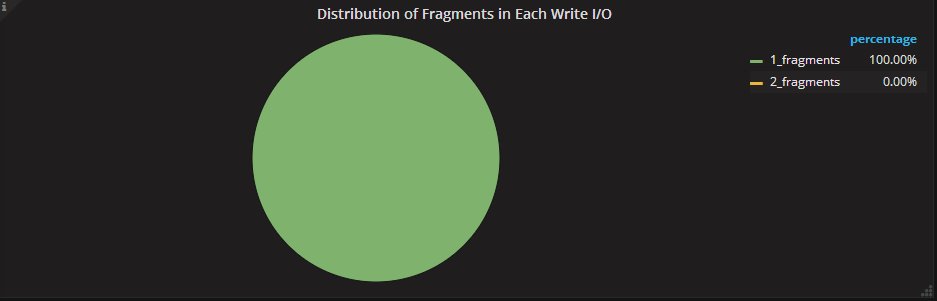
**读I/O不连续块分布面板**(图32) 显示了Lustre文件系统读I/O不连续块的数目比例信息。如图所示，不连续块为“0\_blocks”数目占的块I/O百分比为100%，说明几乎所有的读I/O都没有被分裂，都是连续的。

图32: 读I/O不连续块分布面板



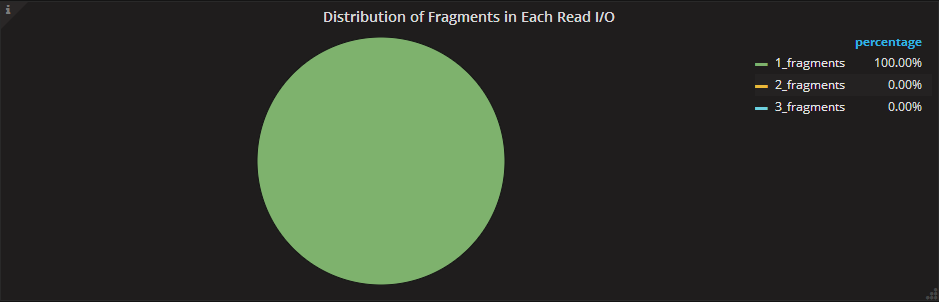
在将读写I/O中提交到磁盘上过程中，由于各种原因（例如单次I/O页面过大），Lustre OSD发起的单个I/O可能被分裂为多个的磁盘I/O。**写I/O碎片分布**面板(图33) 显示了Lustre文件系统每个写I/O分裂成多个磁盘 I/O的数目比例信息。如图所示，碎片为1表示I/O没有被分裂，“1\_fragments”数目占的块I/O百分比为100%，说明几乎所有的写I/O都没有被分裂，都是整块连续的；“2\_fragments”表示Lustre写I/O被分裂为两个磁盘块I/O，其占用的百分比为0%。

图33: 每个写I/O碎片分布面板



**读I/O碎片分布**面板(图34) 显示了Lustre文件系统每个读I/O分裂成碎片化磁盘 I/O的数目比例信息。如图所示，碎片为1表示I/O没有被分裂，“1\_fragments”数目占的块I/O百分比为100%，说明几乎所有的读I/O都没有被分裂碎片化，都是连续的；“2\_fragments”表示Lustre读I/O被分裂为两个磁盘块I/O，其占用的百分比为0%。

图34: 读I/O碎片分布面板



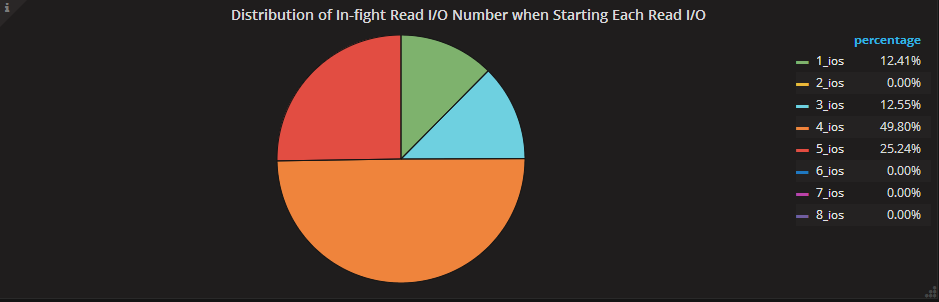
**已提交等待结束的磁盘写I/O分布**面板(图35) 显示了当前OSD已提交等待结束的磁盘写I/O比例信息。如图所示，”1\_ios”表示当前正在运行的磁盘I/O为1个，其所占比例为100%。

图35: 已提交等待结束的磁盘写I/O分布面板



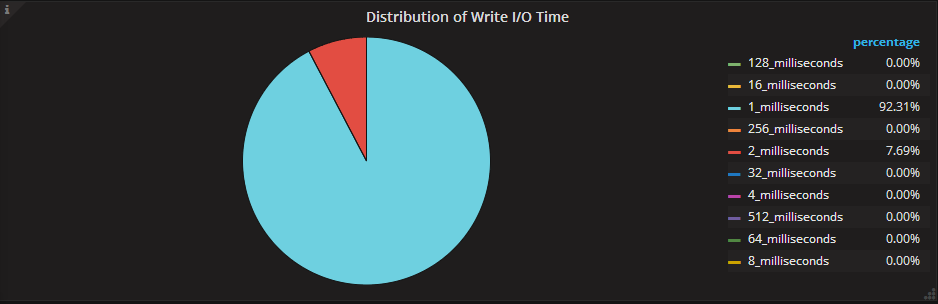
**已提交等待结束的磁盘读I/O分布**面板(图36) 显示了当前OSD已提交等待结束的（正在运行过程中）磁盘读I/O比例信息。如图所示，“1\_ios”表示当前正在运行的磁盘I/O为1个，其所占比例为12.41%；“3\_ios”表示当前挂起的磁盘I/O数目为3，其占的百分比为12.55%；“4\_ios”表示当前挂起的磁盘I/O数目为4，其占的百分位49.80%。

图36: 已提交等待结束的磁盘读I/O分布面板



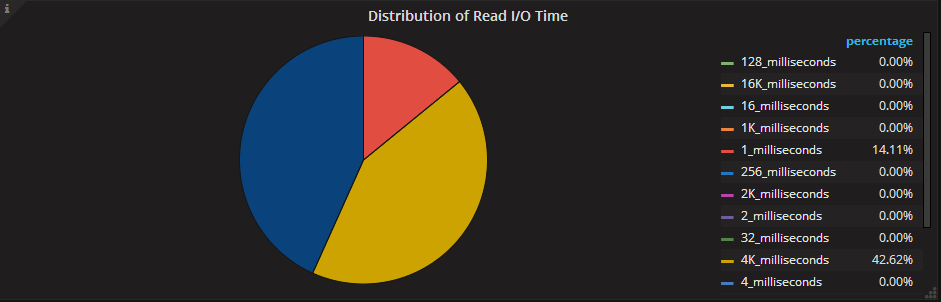
**写I/O时间分布**面板(图37) 显示了当前OSD写I/O时间分布比例信息。如图所示，“1\_milliseconds”表示写I/O时间小于1毫秒的I/O次数所占I/O次数的百分比，其所占比例为92.31%，“2\_milliseconds”表示写I/O时间在1毫秒和2毫秒之间的I/O次数所占百分比，其值为：7.69%。

图37: 写I/O时间分布面板



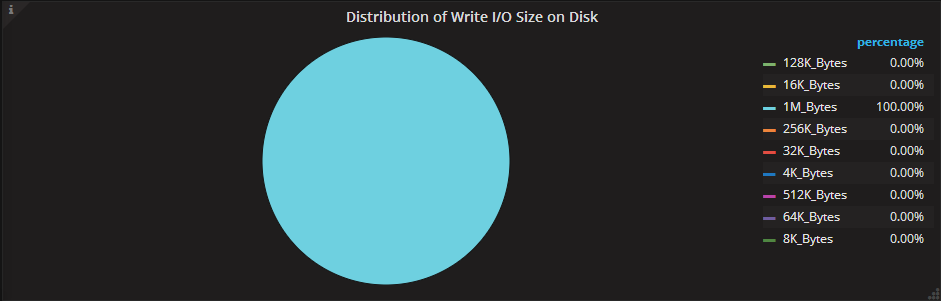
**读I/O时间分布**面板(图38) 显示了OSD读I/O时间分布比例信息。如图所示，“1\_milliseconds”表示读I/O时间小于1毫秒的I/O次数所占I/O次数的百分比，其所占比例为14.11%；“4K\_milliseconds”表示读I/O时间在2K毫秒和4K毫秒之间的I/O次数所占百分比，其值为42.62%。

图38: 读I/O时间分布面板



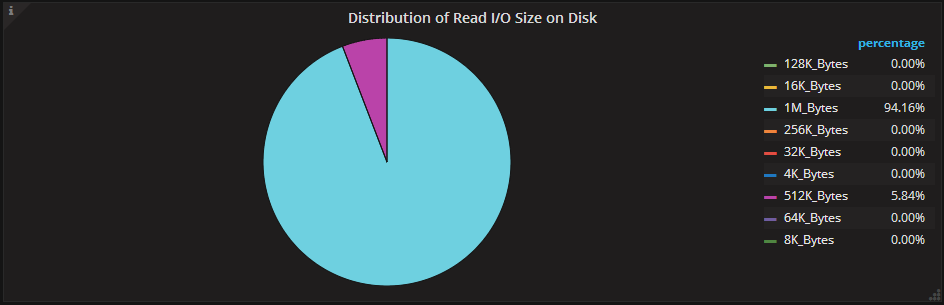
**磁盘写I/O大小分布**面板(图39) 显示了OSD写I/O大小分布比例信息。如图所示，“1M\_Bytes”表示磁盘写I/O大小为512K字节到1M字节之间的I/O次数的所占百分比，其值为100%。“512K \_ Bytes”表示磁盘写I/O大小为256K字节到512K字节之间的I/O次数的所占百分比。

图39: 磁盘写I/O大小分布面板



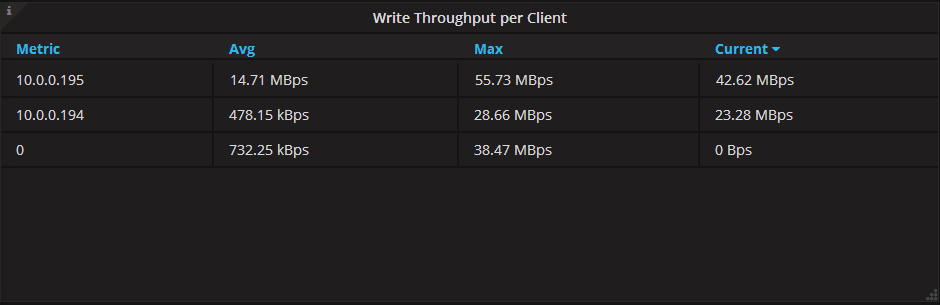
**磁盘读I/O大小分布面板**(图40) 显示了OSD读I/O大小分布比例信息。如图所示，“1M\_Bytes”表示磁盘读I/O大小为512K字节到1M字节之间的I/O次数的所占百分比，其值为94.16%；“512K\_Bytes”表示磁盘读I/O大小在256K字节到512K字节之间的I/O次数所占百分比，其值为5.84%。

图40: 磁盘读I/O大小分布面板



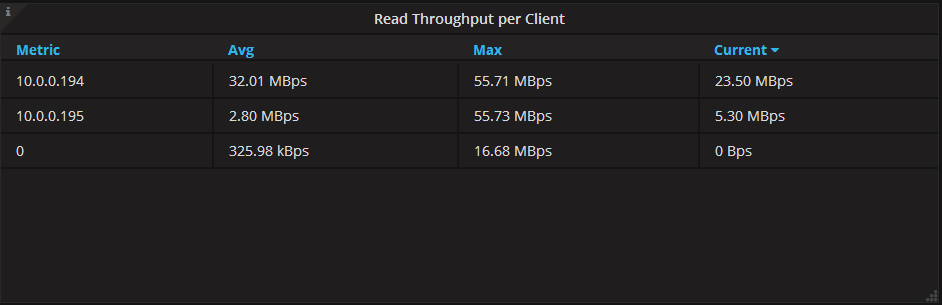
**每个客户端写吞吐量**面板（图41）显示了每个客户端写吞吐量信息，包括平均值，最大值和当前值。图中表示IP地址为“10.0.0.195”的客户端的平均I/O吞吐量为14.71MBps，最大值为55.73MBps，当前吞吐量为42.62MBps。

图41: 每个客户端写吞吐量面板



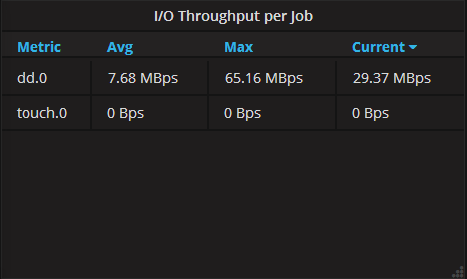
**每个客户端读吞吐量**面板（图42）显示了每个客户端读吞吐量信息，包括平均值，最大值和当前值。图中表示IP地址为“10.0.0.194”的客户端的平均读吞吐量为32.01MBps，最大值为55.71MBps，当前吞吐量为23.50MBps。

图42: 每个客户端读吞吐量面板



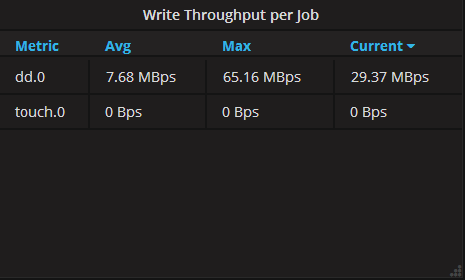
**每个作业I/O吞吐量**面板（图43）显示了每个作业I/O吞吐量信息，包括平均值，最大值和当前值。图中JOBID为“dd.0”的作业的平均I/O吞吐量为7.68MBps，最大值为65.16MBps，当前吞吐量为29.37MBps。

图43: 每个作业I/O吞吐量面板



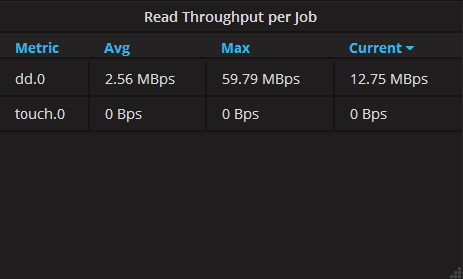
**每个作业写吞吐量**面板（图44）显示了每个作业写吞吐量信息，包括平均值，最大值和当前值。图中JOBID为“dd.0”的作业的平均写吞吐量为7.68MBps，最大值为65.16MBps，当前吞吐量为29.37MBps。

图44: 每个作业写吞吐量面板



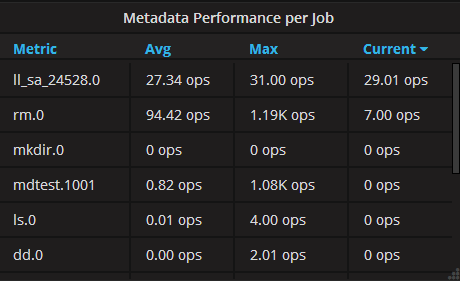
**每个作业读吞吐量**面板（图45）显示了每个作业读吞吐量信息，包括平均值，最大值和当前值。图中JOBID为“dd.0”的作业的平均读吞吐量为2.56MBps，最大值为59.79MBps，当前吞吐量为12.75MBps。

图45: 每个作业读吞吐量面板



**每个作业元数据性能**面板（图46）显示了每个作业元数据性能信息，包括平均值，最大值和当前值，其单位为Ops （Operations per Second）。图中JOBID为“rm.0”的作业的平均元数据性能为94.42ops，最大值为1.19K ops，当前性能为7.00 ops。

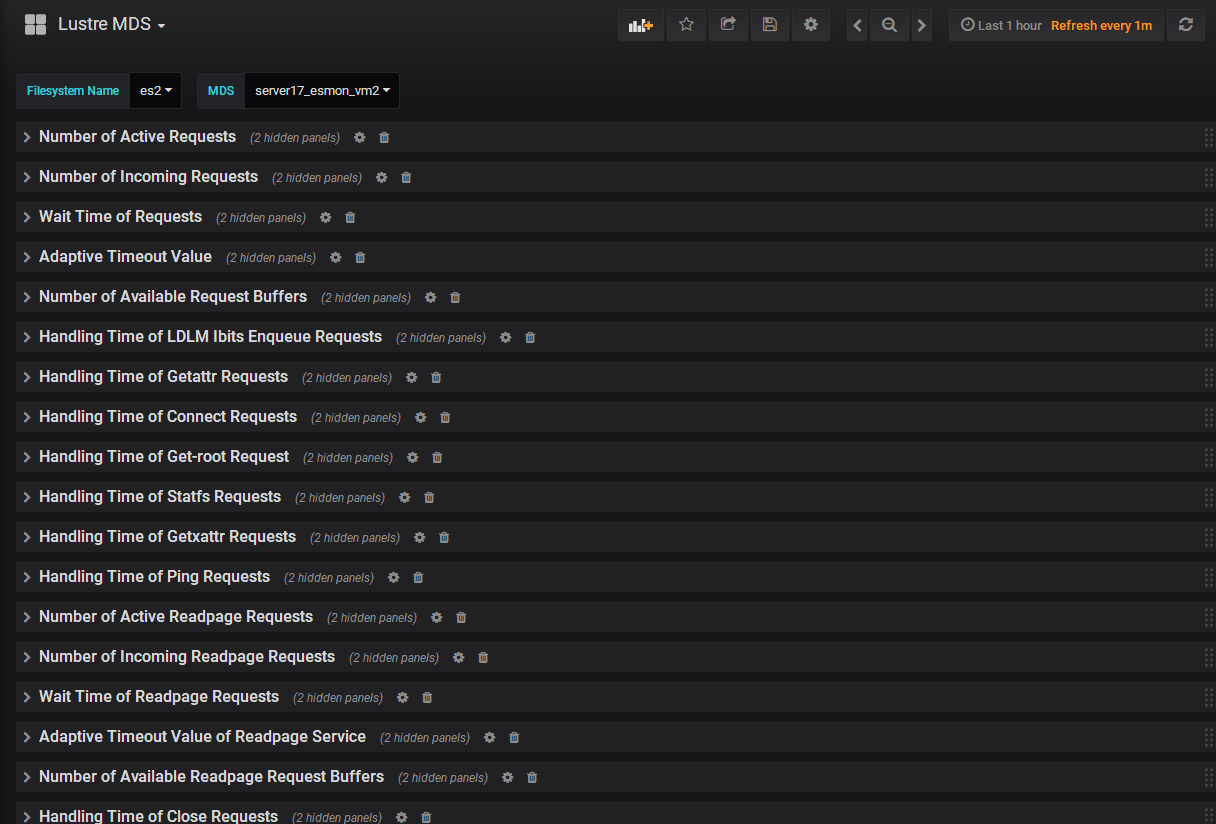
图46: 每个作业元数据性能面板



## Lustre MDS仪表盘

Lustre MDS仪表盘（图47）显示了 Lustre 文件系统MDS服务器的统计数据。

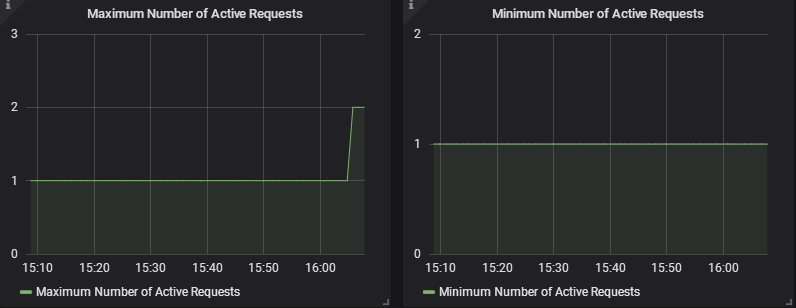
图47: Lustre MDS仪表盘



以下是一些**主要指标**的面板视图：

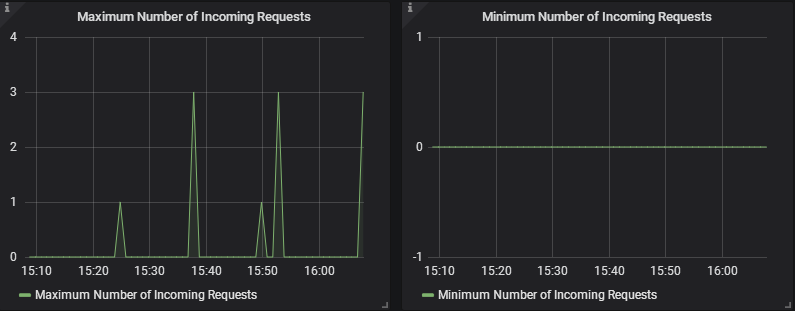
**活跃请求数目**面板（图48）显示分别显示了MDS服务器最大活跃请求数目和最小活跃请求数目随时间的变化。活跃的请求数是指那些正在被MDS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图48：活跃请求数目面板



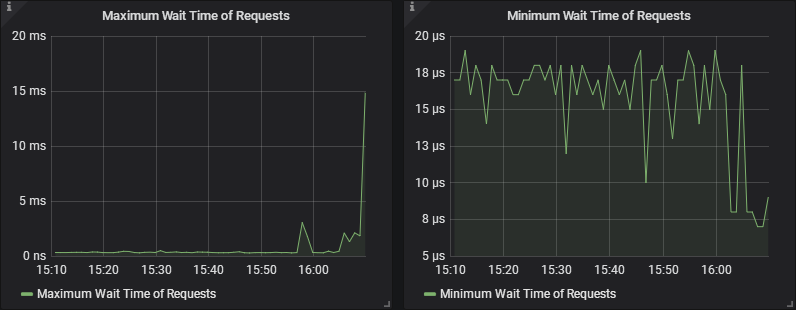
流入请求数目面板（图49）显示分别显示了MDS服务器最大流入请求数目和最小流入请求数目随时间的变化。流入请求是指那些正在等待被MDS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入请求的最大值；而右边的图则显示了收集时间间隔中的流入请求的最小值。

图49：流入请求数目面板



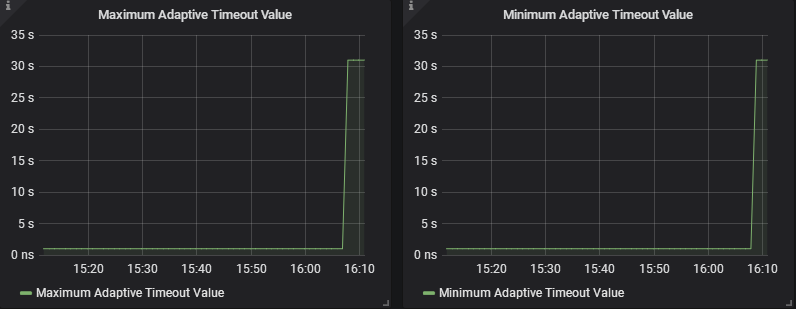
请求等待时间面板（图50）显示分别显示了MDS服务器最大请求等待时间和最小请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的请求等待时间。

图50：请求等待时间面板



自适应超时值面板（图51）分别显示了MDS服务器最大自适应超时值和最小自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的MDS服务的最大超时值；而右边的图则显示了收集时间间隔中的最小值超时值。

图51：自适应超时值面板



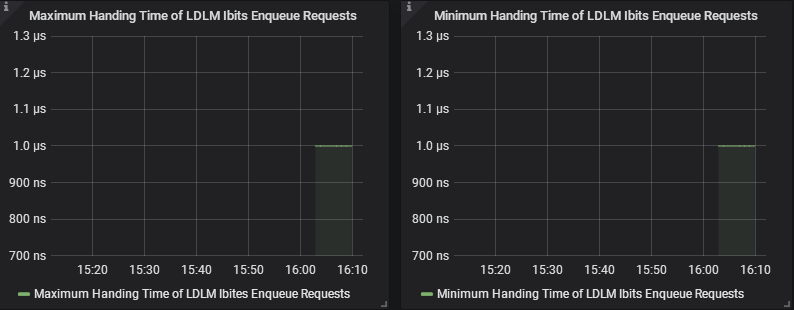
可用请求缓冲区数目面板（图52）分别显示了MDS服务器最大可用请求缓冲区数目和最小可用请求缓冲区数目随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图52：可用请求缓冲区数目面板



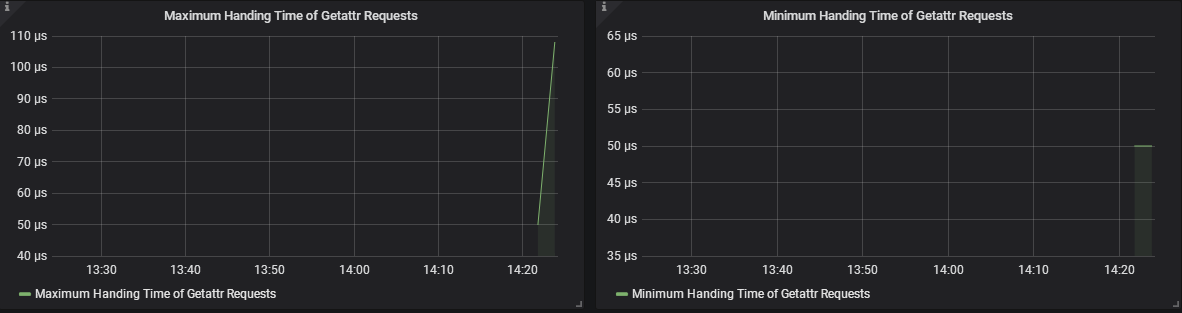
LDLM ibits Enqueue请求处理时间面板（图53）分别显示了MDS服务器最大LDLM ibits Enqueue请求处理时间和最小LDLM ibits Enqueue请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的LDLM ibits Enqueue请求处理时间的最大值；而右边的图则显示了收集时间间隔中的LDLM ibits Enqueue请求处理时间的最小值。

图53：LDLM ibits Enqueue请求处理时间面板



Getattr请求处理时间面板（图54）分别显示了MDS服务器最大Getattr请求处理时间和最小Getattr请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的Getattr请求处理时间的最大值；而右边的图则显示了收集时间间隔中的Getattr请求处理时间的最小值。

图54：Getattr请求处理时间面板



Connect请求处理时间面板（图55）分别显示了MDS服务器最大Connect请求处理时间和最小Connect请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的Connect请求处理时间的最大值；而右边的图则显示了收集时间间隔中的Connect请求处理时间的最小值。

图55：Connect请求处理时间面板



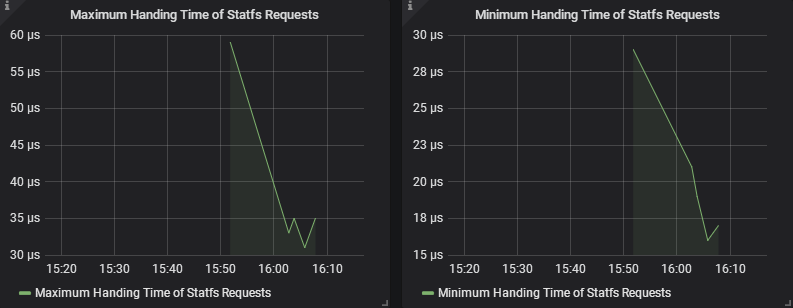
Get-root请求处理时间面板（图56）分别显示了MDS服务器最大Get-root请求处理时间和最小Get-root请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的Get-root请求处理时间的最大值；而右边的图则显示了收集时间间隔中的Get-root请求处理时间的最小值。

图56：Get-root请求处理时间面板



Statfs请求处理时间面板（图57）分别显示了MDS服务器最大statfs请求处理时间和最小statfs请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的statfs请求处理时间的最大值；而右边的图则显示了收集时间间隔中的statfs请求处理时间的最小值。

图57：Statfs请求处理时间面板



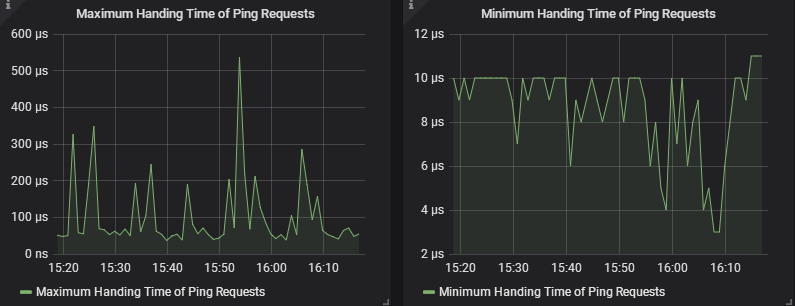
Getxattr请求处理时间面板（图58）分别显示了MDS服务器最大Getxattr请求处理时间和最小Getxattr请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的Getxattr请求处理时间的最大值；而右边的图则显示了收集时间间隔中的Getxattr请求处理时间的最小值。

图58：Getxattr请求处理时间面板



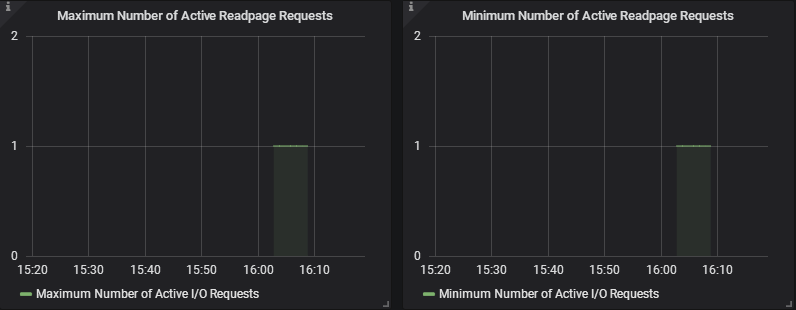
Ping请求处理时间面板（图59）分别显示了MDS服务器最大ping请求处理时间和最小ping请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的ping请求处理时间的最大值；而右边的图则显示了收集时间间隔中的ping请求处理时间的最小值。

图59：Ping请求处理时间面板



**活跃的Readpage请求数目**面板（图60）分别显示了MDS服务器最大活跃的Readpage请求数目和最小活跃的Readpage请求数目随时间的变化。活跃的请求数是指那些正在被MDS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图60：活跃的Readpage请求数目面板



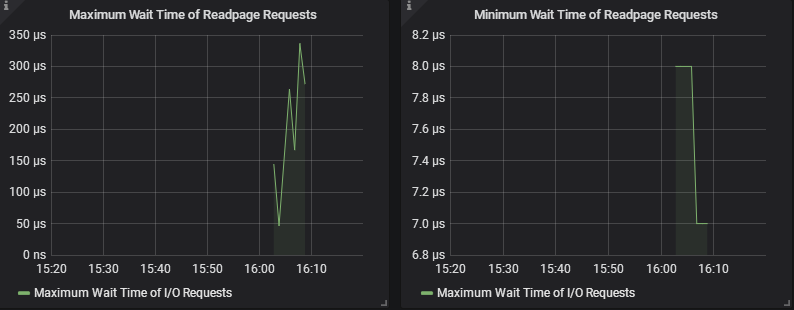
流入的Readpage请求数目面板（图61）分别显示了MDS服务器最大流入的Readpage请求数目和最小流入的Readpage请求数目随时间的变化。流入请求是指那些正在等待被MDS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入请求的最大值；而右边的图则显示了收集时间间隔中的最小值。

图61：流入的Readpage请求数目面板



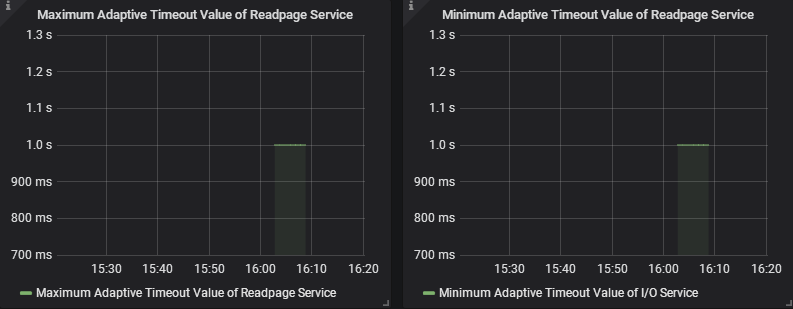
Readpage请求等待时间面板（图62）分别显示了MDS服务器最大Readpage请求等待时间和最小Readpage请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的请求等待时间。

图62：Readpage请求等待时间面板



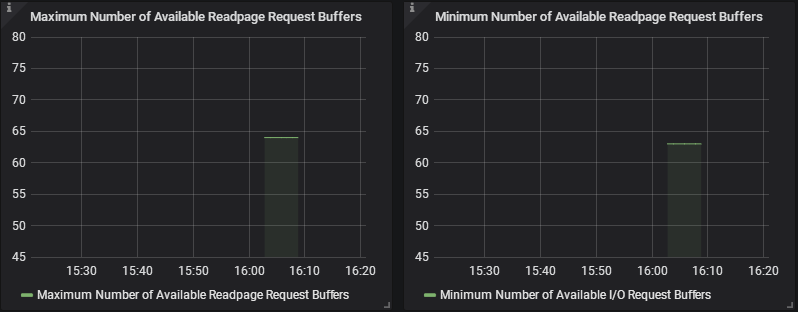
Readpage服务的自适应超时值面板（图63）分别显示了MDS服务器最大Readpage服务的自适应超时值和最小Readpage服务的自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的MDS Readpage服务的最大超时值；而右边的图则显示了收集时间间隔中的最小值超时值。

图63：Readpage自适应超时值面板



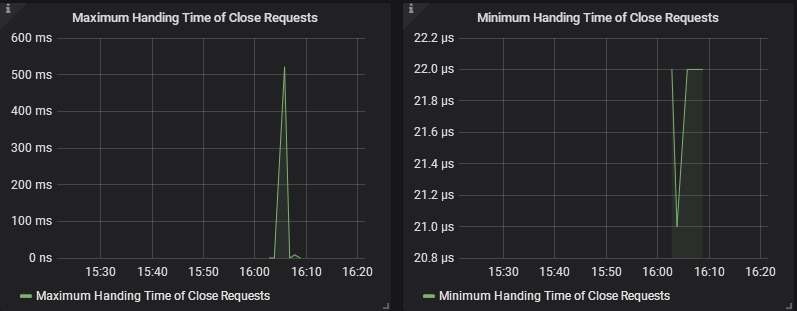
可用Readpage请求缓冲区数目面板（图64）分别显示了MDS服务器最大可用Readpage请求缓冲区数目和最小可用Readpage请求缓冲区数目随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图64：可用Readpage请求缓冲区数目面板



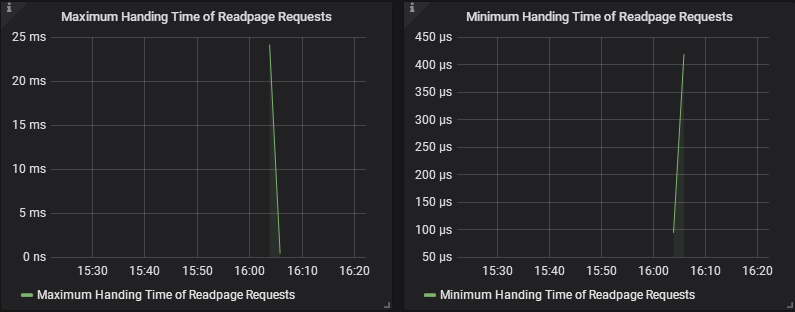
Close请求处理时间面板（图65）分别显示了MDS服务器最大Close请求处理时间和最小Close请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的Close请求处理时间的最大值；而右边的图则显示了收集时间间隔中的Close请求处理时间的最小值。

图65：Close请求处理时间面板



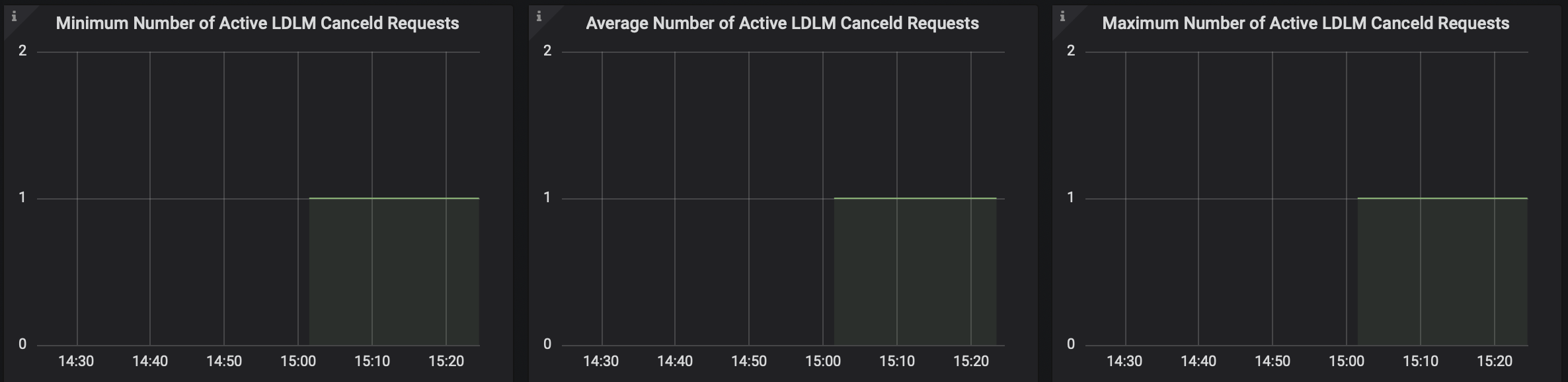
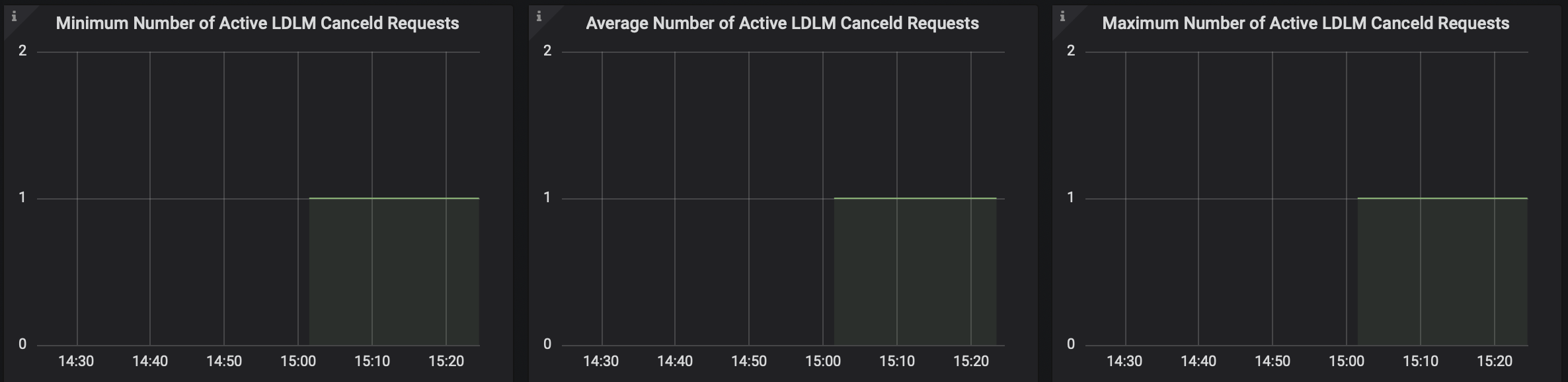
Readpage请求处理时间面板（图66）分别显示了MDS服务器最大Readpage请求处理时间和最小Readpage请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的Readpage请求处理时间的最大值；而右边的图则显示了收集时间间隔中的Readpage请求处理时间的最小值。

图66：Readpage请求处理时间面板



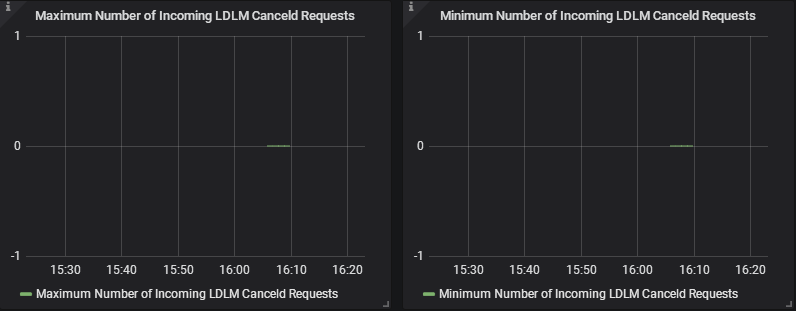
**活跃的LDLM Canceld请求数目**面板（图67）分别显示了MDS服务器最大活跃的LDLM Canceld请求数目和最小活跃的LDLM Canceld请求数目随时间的变化。活跃的请求数是指那些正在被MDS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图67：活跃的LDLM Canceld请求数目面板



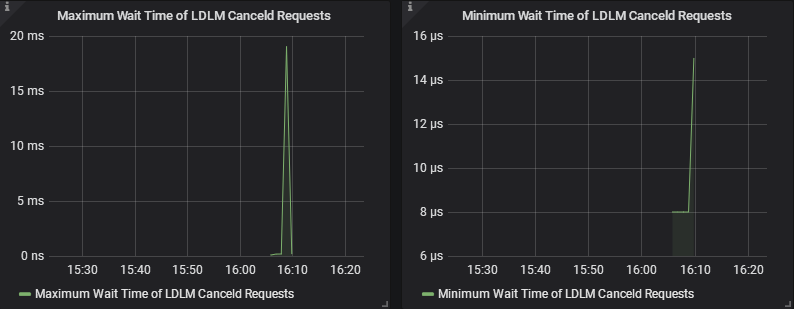
流入的LDLM Canceld请求数目面板（图68）分别显示了MDS服务器最大流入的LDLM Canceld请求数目和最小流入的LDLM Canceld请求数目随时间的变化。流入请求是指那些正在等待被MDS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入请求的最大值；而右边的图则显示了收集时间间隔中的最小值。

图68：流入的LDLM Canceld请求数目面板



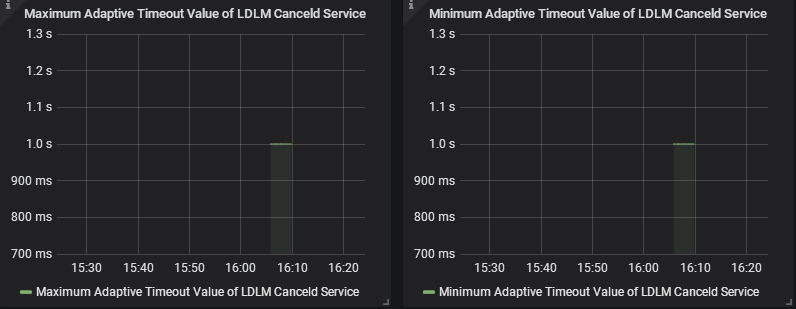
LDLM Canceld请求等待时间面板（图69）分别显示了MDS服务器最大LDLM Canceld请求等待时间和最小LDLM Canceld请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的请求等待时间。

图69：LDLM Canceld请求等待时间面板



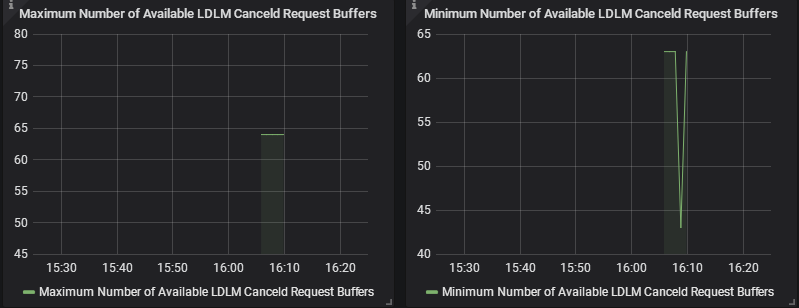
LDLM Canceld服务的自适应超时值面板（图70）分别显示了MDS服务器最大LDLM Canceld服务的自适应超时值和最小LDLM Canceld服务的自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的最大的LDLM Canceld服务的自适应超时值；而右边的图则显示了上一个收集时间间隔中的最小的LDLM Canceld服务的自适应超时值。

图70：LDLM Canceld服务的自适应超时值面板



可用LDLM Canceld请求缓冲区数目面板（图71）分别显示了MDS服务器最大可用LDLM Canceld请求缓冲区数目和最小可用LDLM Canceld请求缓冲区数目随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图71：可用LDLM Canceld请求缓冲区数目面板



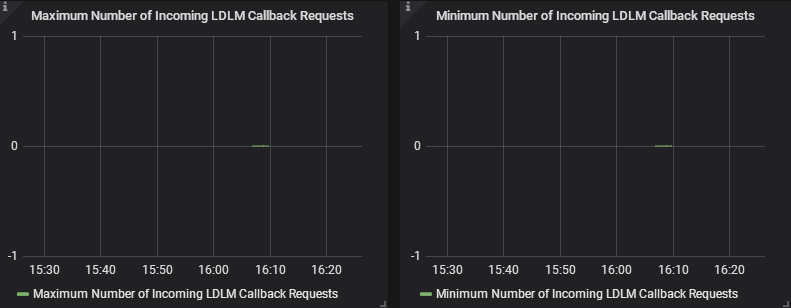
活跃的LDLM Callback请求数目面板（图72）分别显示了MDS服务器最大活跃的LDLM Callback请求数目和最小活跃的LDLM Callback请求数目随时间的变化。活跃的请求数是指那些正在被MDS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图72：活跃的LDLM Callback请求数目面板



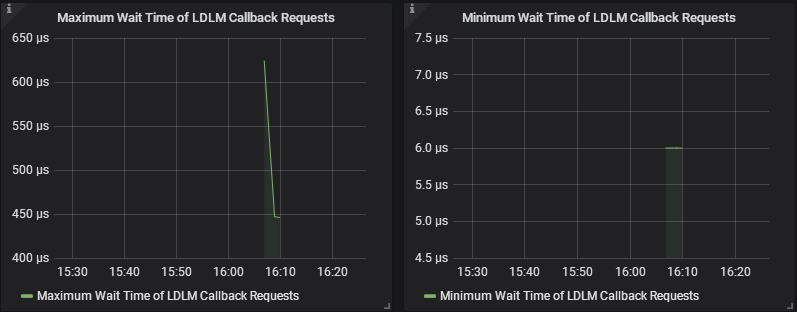
流入的LDLM Callback请求数目面板（图73）分别显示了MDS服务器最大流入的LDLM Callback请求数目和最小流入的LDLM Callback请求数目随时间的变化。流入请求是指那些正在等待被MDS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入请求的最大值；而右边的图则显示了收集时间间隔中的最小值。

图73：流入的LDLM Callback请求数目面板



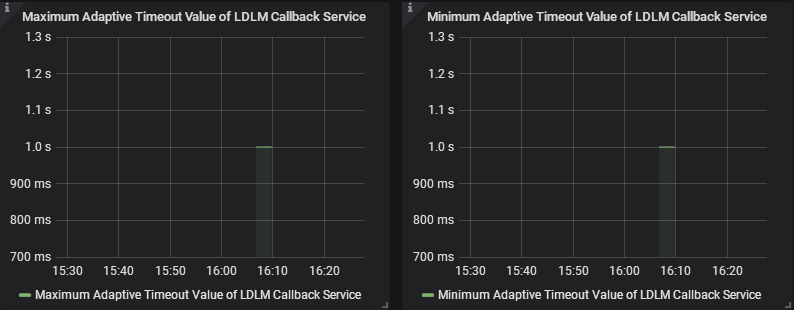
LDLM Callback请求等待时间面板（图74）分别显示了MDS服务器最大LDLM Callback请求等待时间和最小LDLM Callback请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的请求等待时间。

图74：LDLM Callback请求等待时间面板



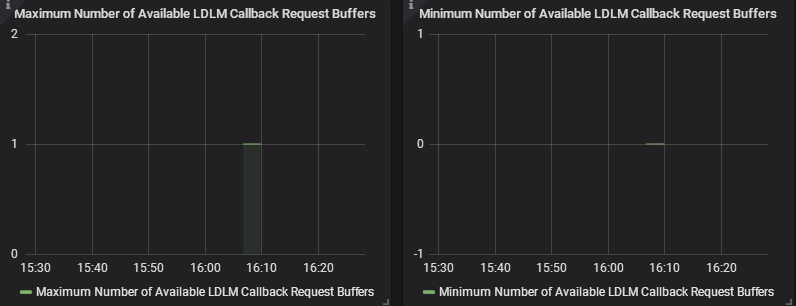
LDLM Callback服务的自适应超时值面板（图75）分别显示了MDS服务器最大LDLM Callback服务的自适应超时值和最小LDLM Callback服务的自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的MDS LDLM Callback服务的最大超时值；而右边的图则显示了收集时间间隔中的最小值超时值。

图75：LDLM Callback自适应超时值面板



可用LDLM Callback请求缓冲区数目面板（图76）分别显示了MDS服务器最大可用LDLM Callback请求缓冲区数目和最小可用LDLM Callback请求缓冲区数目随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

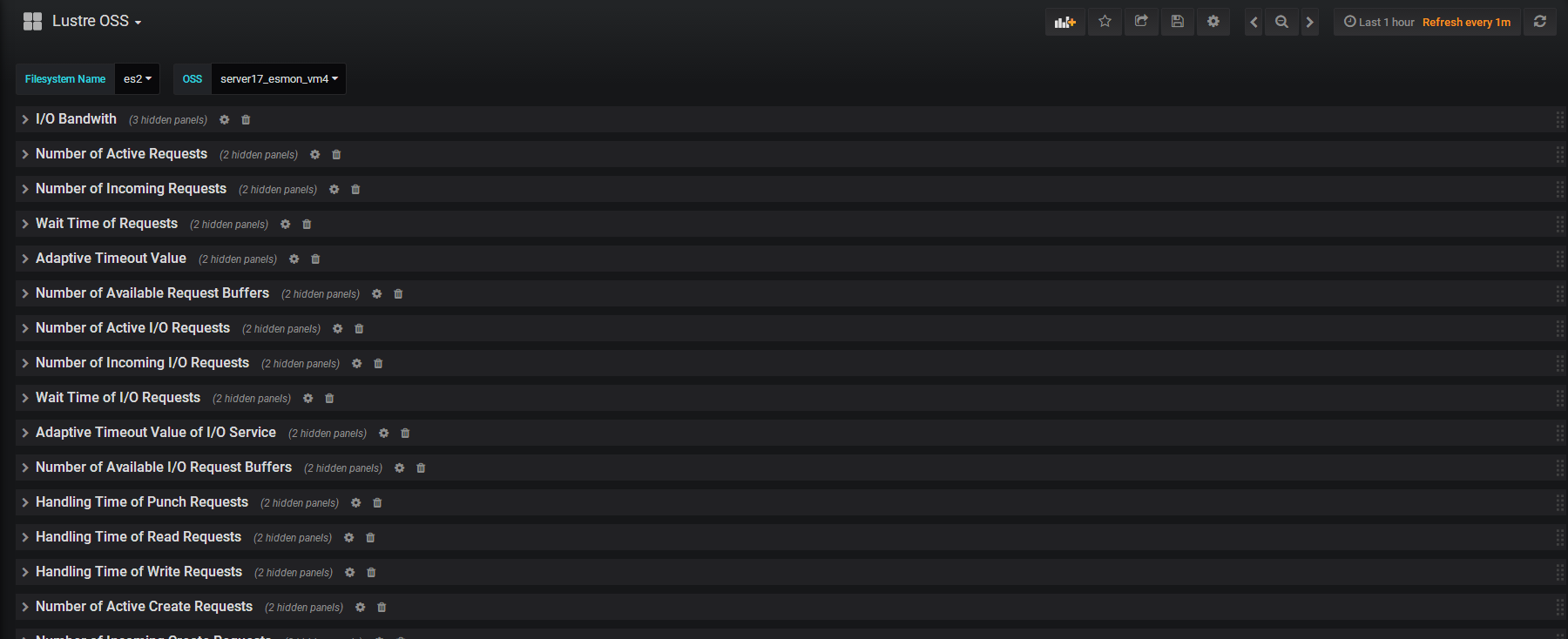
图76：可用LDLM Callback请求缓冲区数目面板



## Lustre OSS仪表盘

Lustre OSS仪表盘（图77: 图77）显示了 Lustre 文件系统OSS服务器的统计数据。

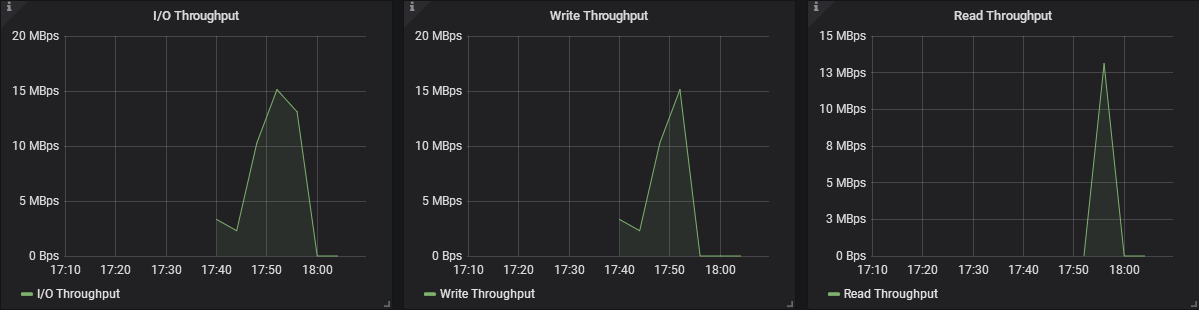
图77: Lustre OSS仪表盘



以下是一些**主要指标**的面板视图：

I/O带宽面板（图78）显示分别显示了OSS服务器总的I/O吐吞量，写吞吐量和读吞吐量。

图78：I/O带宽面板



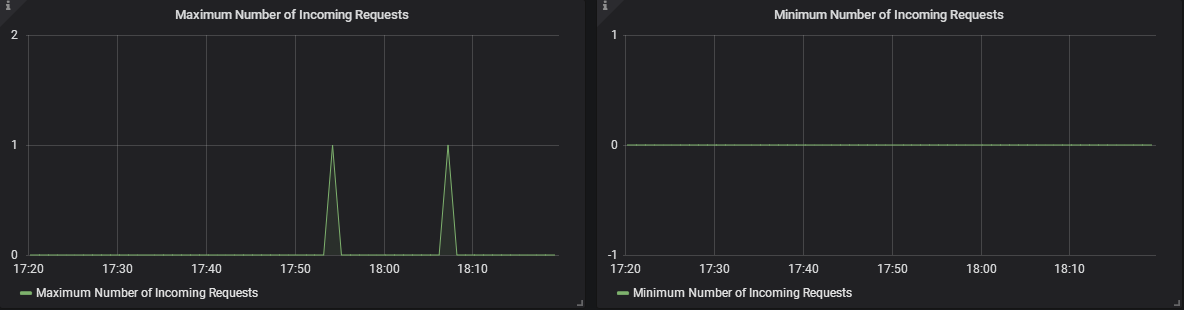
**活跃请求数目**面板（图79）显示了OSS最大活跃请求数目和最小活跃请求数目随时间的变化。活跃的请求数是指那些正在被OSS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图79: 活跃请求数目面板



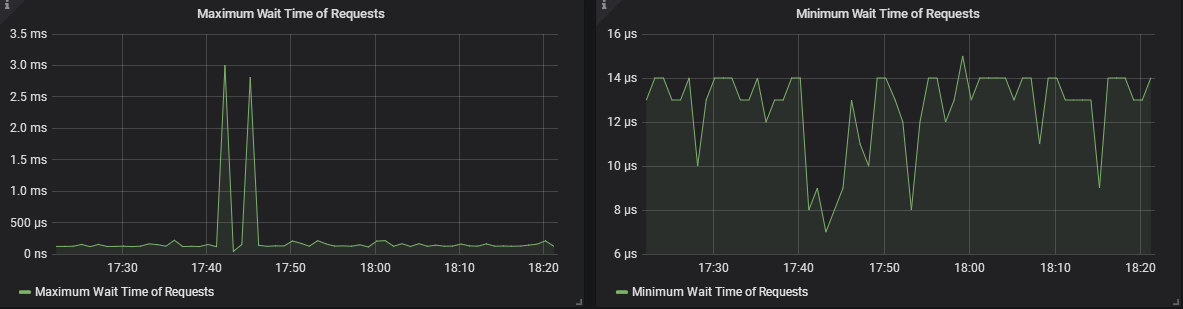
流入请求数目面板（图80）显示了OSS最大流入请求数目和最小流入请求数目随时间的变化。流入请求是指那些正在等待被OSS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入请求的最大值；而右边的图则显示了收集时间间隔中的最小值。

图80: 流入请求数目面板



请求等待时间面板（图81）显示了OSS最大请求等待时间和最小请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的请求等待时间。

图81: 请求等待时间面板



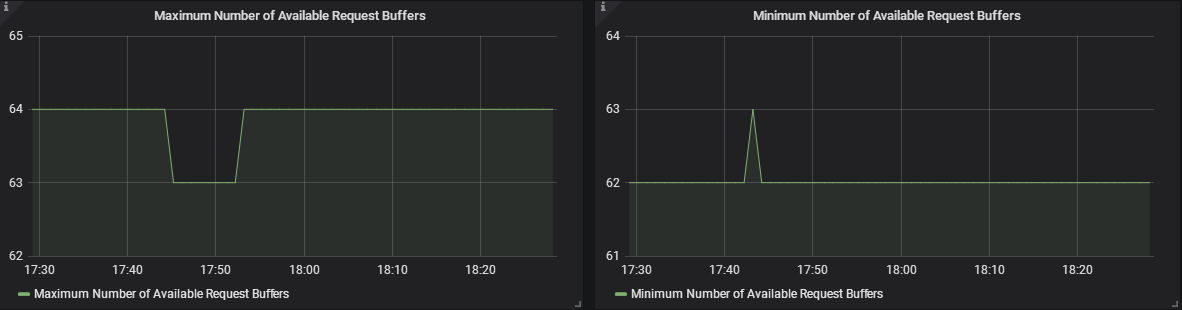
自适应超时值面板（图82）显示了OSS最大自适应超时值和最小自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的OSS服务的最大超时值；而右边的图则显示了收集时间间隔中的最小超时值。

图82: 自适应超时值面板



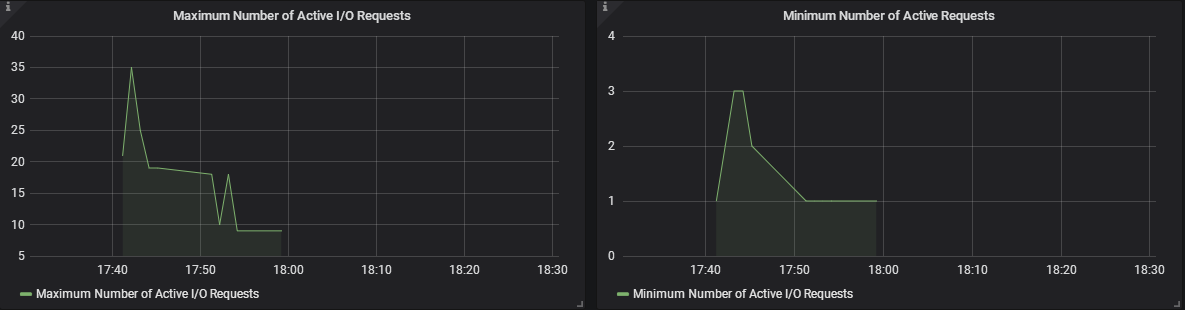
可用请求缓冲区数目面板（图83）显示了OSS最大可用请求缓冲区数目和最小可用请求缓冲区数目随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图83: 可用请求缓冲区数目面板



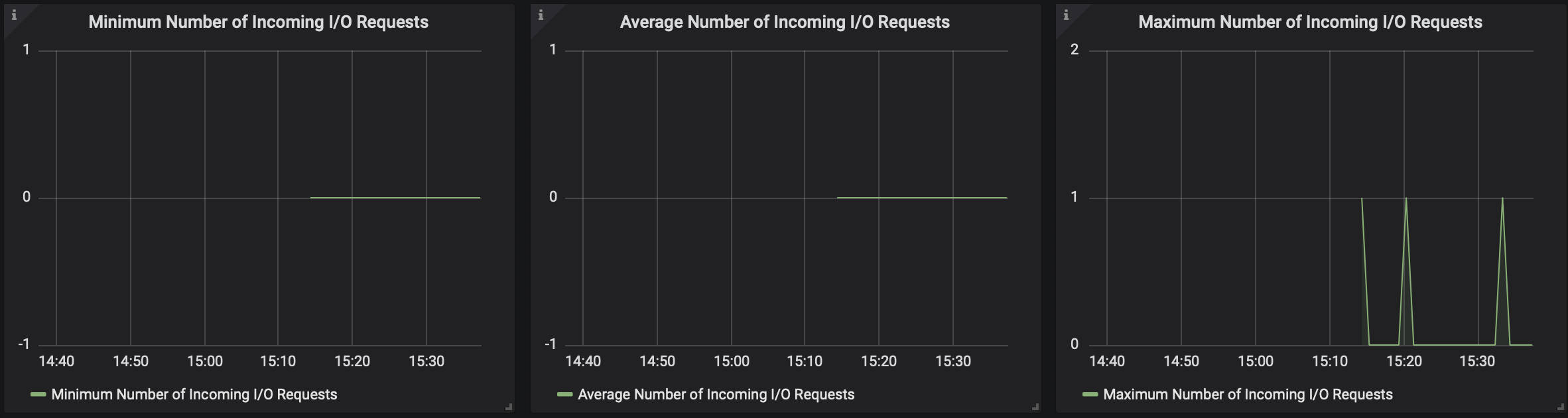
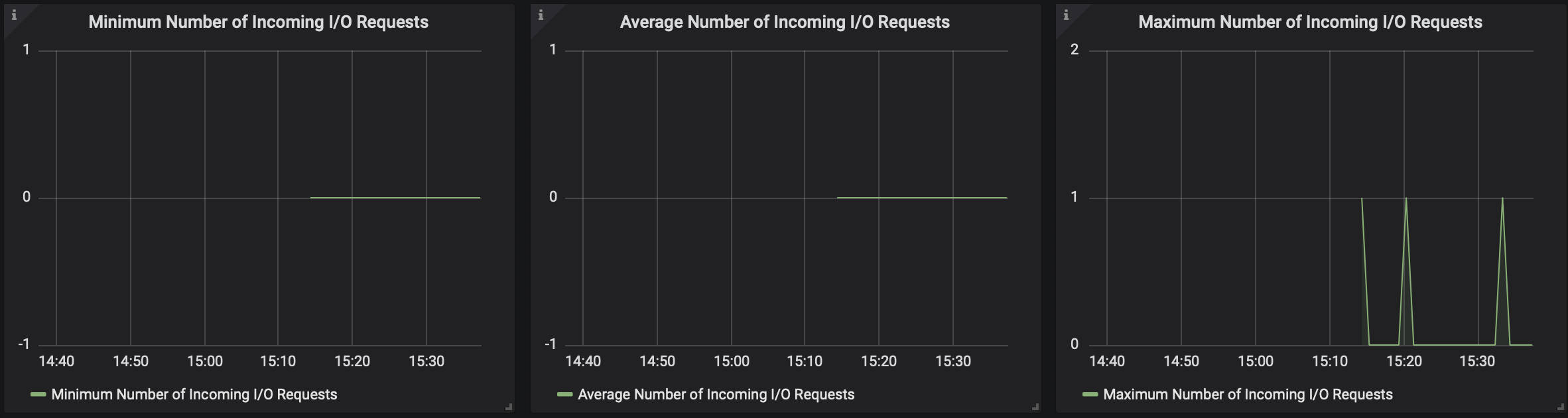
**活跃的I/O请求数目**面板（图84）显示了OSS最大活跃的I/O请求数目和最小活跃的I/O请求数目随时间的变化。活跃的请求数是指那些正在被OSS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图84: 活跃的I/O请求数目面板



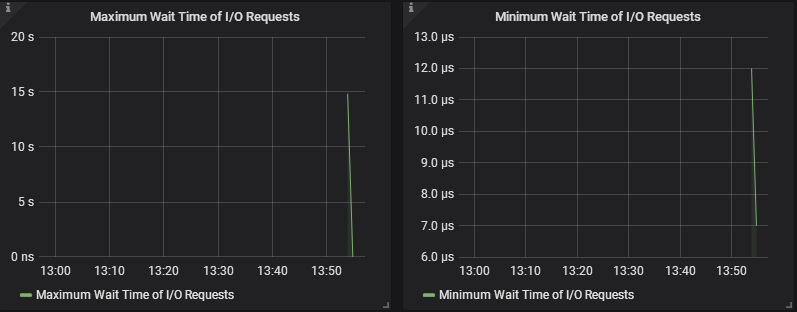
流入I/O请求数目面板（图85）显示了OSS最大流入的I/O请求数目和最小流入的I/O请求数目随时间的变化。流入请求是指那些正在等待被OSS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入I/O请求的最大值；而右边的图则显示了收集时间间隔中的流入I/O请求数目的最小值。

图85: 流入的I/O请求数目面板



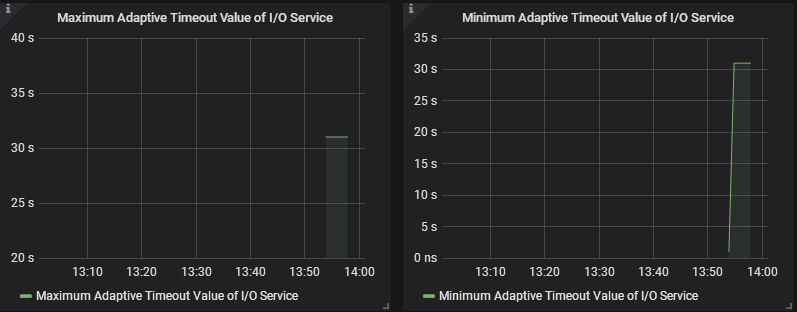
I/O请求等待时间面板（图86）显示了OSS最大I/O请求等待时间和最小I/O请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的I/O请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的I/O请求等待时间。

图86: I/O请求等待时间面板



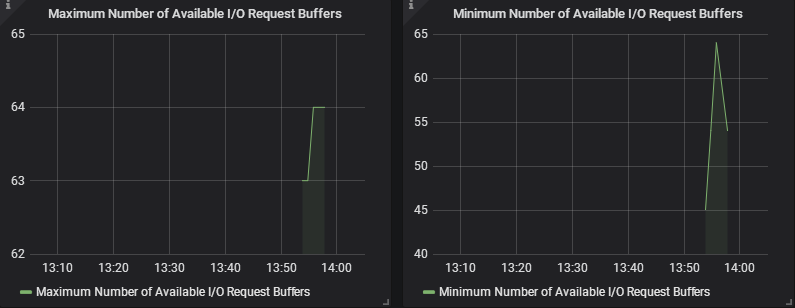
I/O服务的自适应超时值面板（图87）显示了OSS最大I/O服务的自适应超时值和I/O服务的自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的OSS I/O服务的最大超时值；而右边的图则显示了收集时间间隔中的OSS I/O服务的最小超时值。

图87: I/O服务的自适应超时值面板



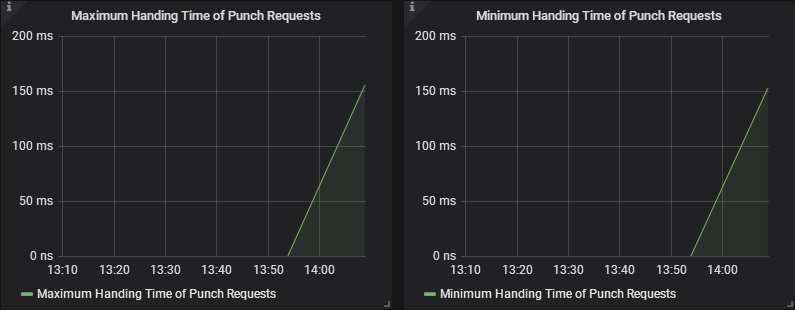
可用I/O请求缓冲区数目面板（图88）显示了OSS最大I/O请求等待时间和最小I/O请求等待时间随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图88: 可用I/O请求缓冲区数目面板



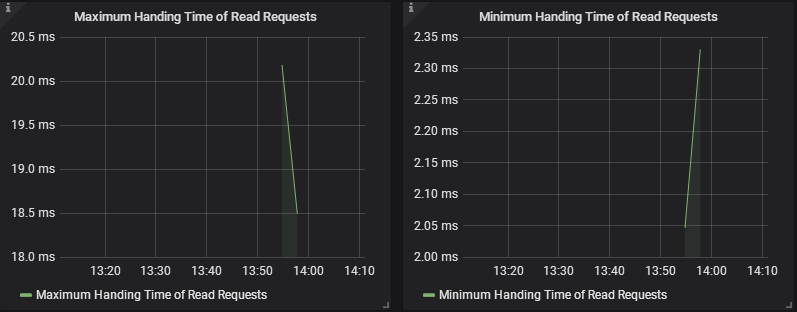
Punch请求处理时间面板（图89）显示了OSS最大Punch请求处理时间和最小Punch请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的Punch请求处理时间的最大值；而右边的图则显示了收集时间间隔中的Punch请求处理时间的最小值。

图89: Punch请求处理时间面板



读请求处理时间面板（图90）显示了OSS最大读请求处理时间和最小读请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的读请求处理时间的最大值；而右边的图则显示了收集时间间隔中的读请求处理时间的最小值。

图90: 读请求处理时间面板



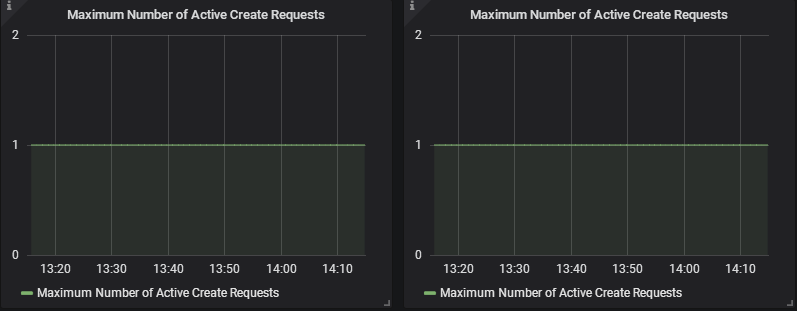
写请求处理时间面板（图91）显示了OSS最大写请求处理时间和最小写请求处理时间随时间的变化。请求的处理时间是指它开始被处理的时刻和处理完成时刻之间的时间间隔。下面左边的图显示了在上一个收集时间间隔中的写请求处理时间的最大值；而右边的图则显示了收集时间间隔中的写请求处理时间的最小值。

图91: 写请求处理时间面板



**活跃的创建请求数目**面板（图92）显示了OSS最大活跃的创建请求数目和最小活跃的创建请求数目随时间的变化。活跃的请求数是指那些正在被OSS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图92: 活跃的创建请求数目面板



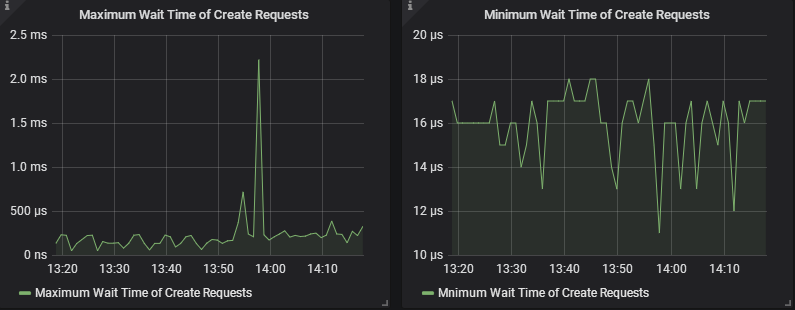
流入的创建请求数目面板（图93）显示了OSS最大流入的创建请求数目和流入的创建请求数目随时间的变化。流入请求是指那些正在等待被OSS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入创建请求的最大值；而右边的图则显示了收集时间间隔中的最小值。

图93: 流入的创建请求数目面板



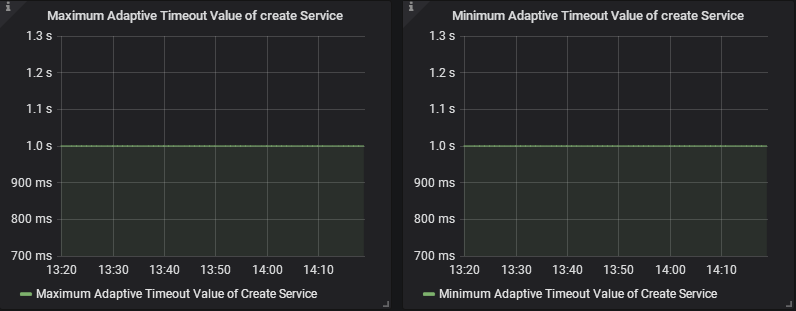
创建请求等待时间面板（图94）显示了OSS最大创建请求等待时间和最小创建请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的创建请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的创建请求等待时间。

图94: 创建请求等待时间面板



创建服务的自适应超时值面板（图95）显示了OSS最大创建服务的自适应超时值和最小创建服务的自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的OSS创建服务的最大超时值；而右边的图则显示了收集时间间隔中的最小值超时值。

图95: 创建服务的自适应超时值面板



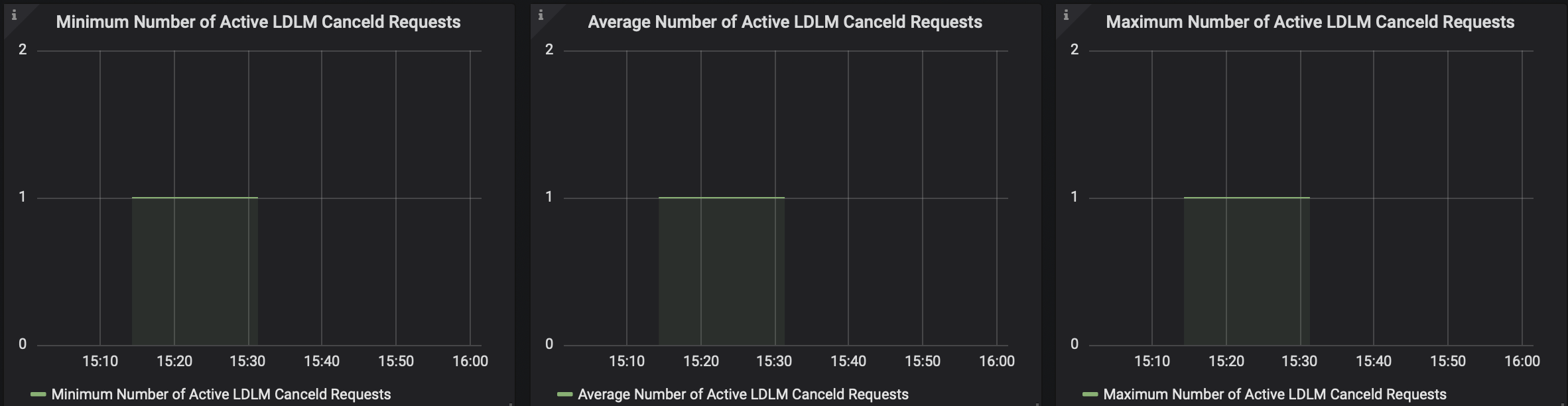
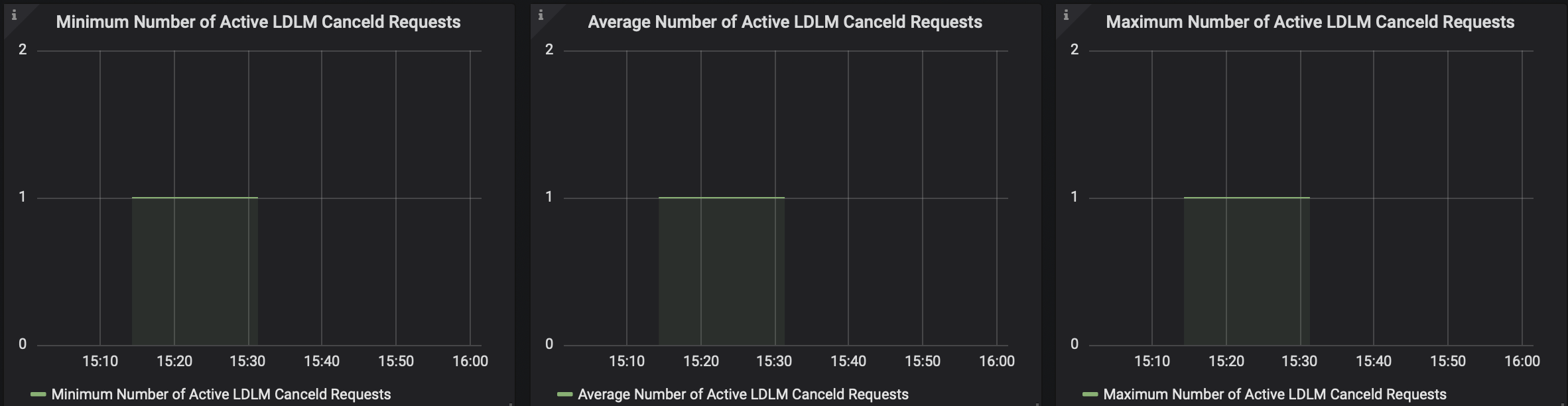
可用创建请求缓冲区数目面板（图96）显示了OSS最大可用创建请求缓冲区数目和可用创建请求缓冲区数目随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图96: 可用创建请求缓冲区数目面板



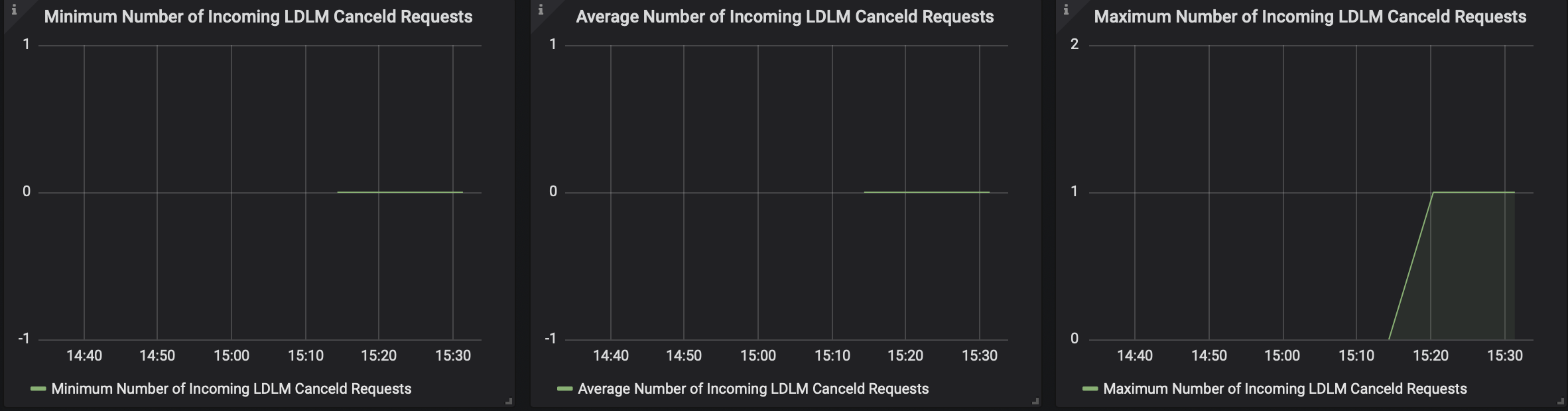
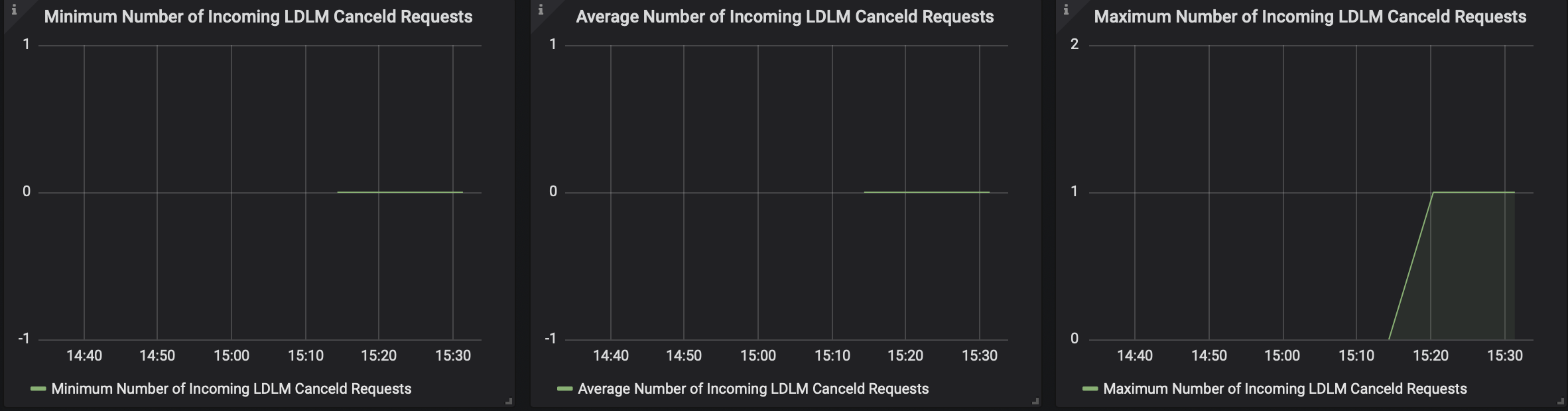
**活跃的LDLM Canceld请求数目**面板（图97）显示了OSS最大活跃的LDLM Canceld请求数目和最小活跃的LDLM Canceld请求数目随时间的变化。活跃的请求数是指那些正在被OSS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图97: 活跃的LDLM Canceld请求数目面板



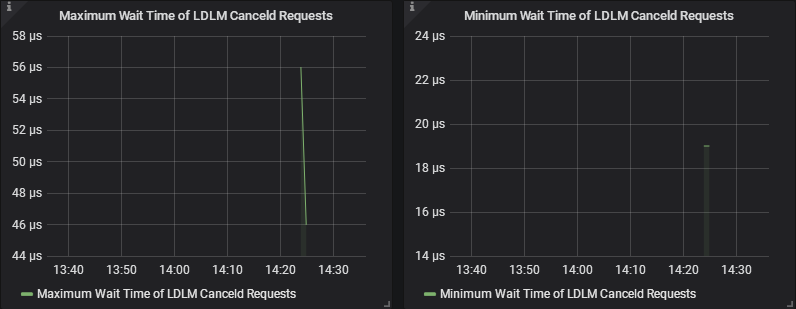
流入的LDLM Canceld请求数目面板（图98）显示了OSS最大流入的LDLM Canceld请求数目和最小流入的LDLM Canceld请求数目随时间的变化。流入请求是指那些正在等待被OSS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入请求的最大值；而右边的图则显示了收集时间间隔中的最小值。

图98: 流入的LDLM Canceld请求数目面板



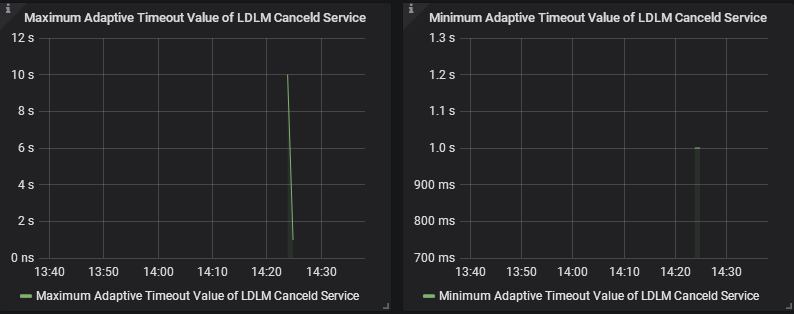
LDLM Canceld请求等待时间面板（图99）显示了OSS最大LDLM Canceld请求等待时间和最小LDLM Canceld请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的LDLM Canceld请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的LDLM Canceld请求等待时间。

图99: LDLM Canceld请求等待时间面板



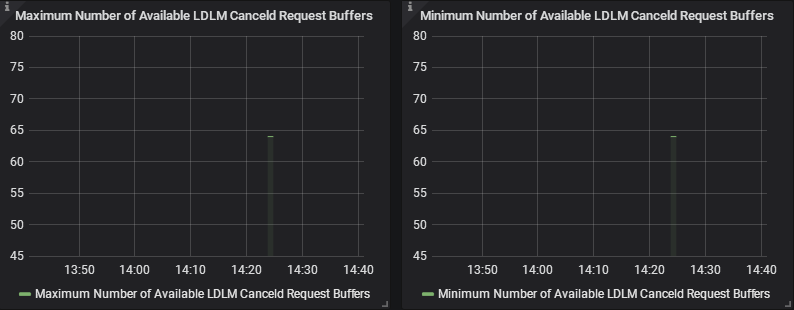
LDLM Canceld服务自适应超时值面板（图100）显示了OSS最大LDLM Canceld服务自适应超时值和最小LDLM Canceld服务自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的OSS LDLM Canceld服务的最大超时值；而右边的图则显示了收集时间间隔中的最小值超时值。

图100: LDLM Canceld服务自适应超时值面板



可用的LDLM Canceld请求缓冲区数目面板（图101）显示了OSS最大可用的LDLM Canceld请求缓冲区数目和最小可用的LDLM Canceld请求缓冲区数目随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图101: 可用的LDLM Canceld请求缓冲区数目面板



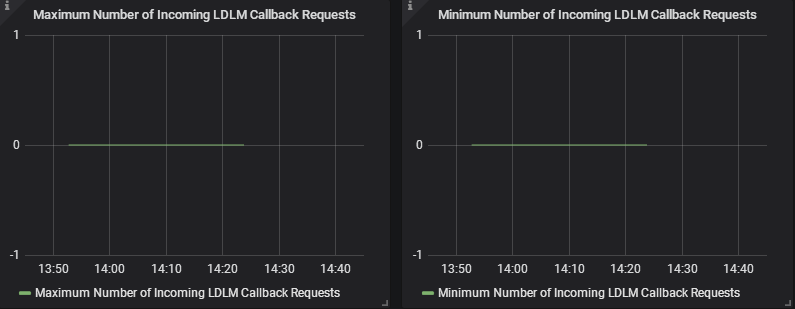
**活跃的LDLM Callback请求数目**面板（图102）显示了OSS最大活跃的LDLM Callback请求数目和最小活跃的LDLM Callback请求数目随时间的变化。活跃的请求数是指那些正在被OSS处理的请求数目，但不包括那些在队列里面等待被处理的请求。如果活跃的请求的数目小于PTLRPC线程数目减去2（一个用于流入请求处理，另一个用于高优先级请求处理），则通常表示线程数目是足够的。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

图102: 活跃的LDLM Callback请求数目面板



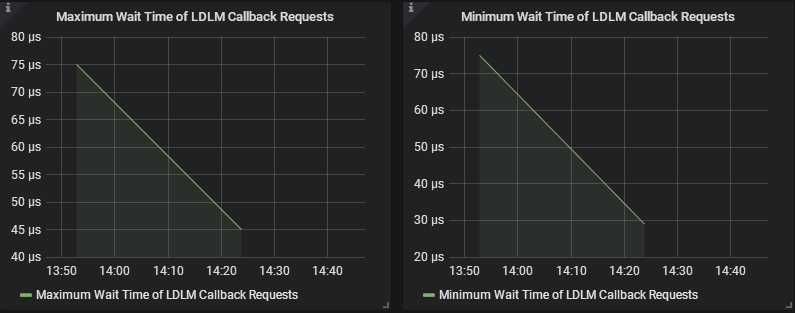
流入的LDLM Callback请求数目面板（图103）显示了OSS最大流入的LDLM Callback请求数目和最小流入的LDLM Callback请求数目随时间的变化。流入请求是指那些正在等待被OSS处理的请求。当开始处理请求，则该请求不再是流入请求，请求将会被放在处理队列中。下面左边的图显示了在上一个收集时间间隔中的流入请求的最大值；而右边的图则显示了收集时间间隔中的最小值。

图103: 流入的LDLM Callback请求数目面板



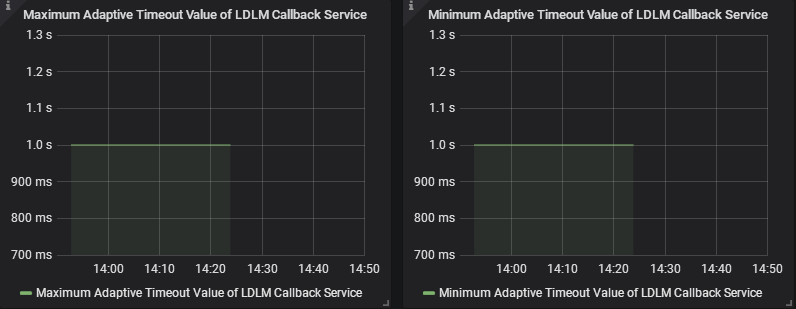
LDLM Callback请求等待时间面板（图104）显示了OSS最大LDLM Callback请求等待时间和最小LDLM Callback请求等待时间随时间的变化。一个请求的等待时间是它到达时刻和开始被处理时刻的时间间隔。下面左边的图显示了在上一个收集时间间隔中的最大的请求等待时间；而右边的图则显示了上一个收集时间间隔中的最小的请求等待时间。

图104: LDLM Callback请求等待时间面板



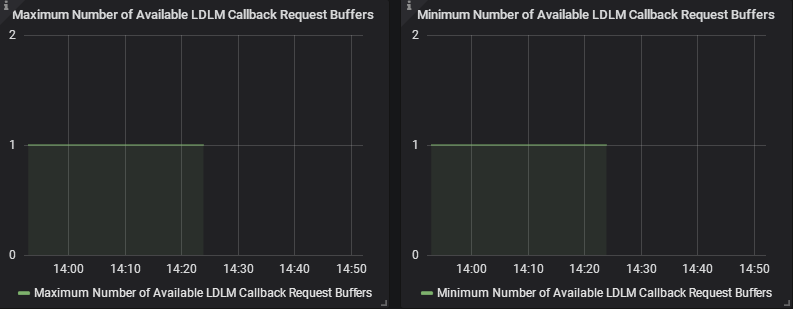
LDLM Callback服务自适应超时值面板（图105）显示了OSS最大LDLM Callback服务自适应超时值和最小LDLM Callback服务自适应超时值随时间的变化。当一个客户端发送请求时，它会给请求的回复一个超时截止时间。由于Lustre的自适应超时机制，服务的超时值是一个运行时服务器和客户端协商的自适应的值。下面左边的图显示了在上一个收集时间间隔中的OSS LDLM Callback服务的最大超时值；而右边的图则显示了收集时间间隔中的最小值超时值。

图105: LDLM Callback服务自适应超时值面板



可用的LDLM Callback请求缓冲区数目面板（图106）显示了OSS最大可用的LDLM Callback请求缓冲区数目和最小可用的LDLM Callback请求缓冲区数目随时间的变化。当一个请求到达时，它会使用一个请求缓冲区。当可用的请求缓冲区数目处于一个低阈值时，就需要更多的缓冲区来避免成为性能的瓶颈。下面左边的图显示了在上一个收集时间间隔中的最大值；而右边的图则显示了收集时间间隔中的最小值。

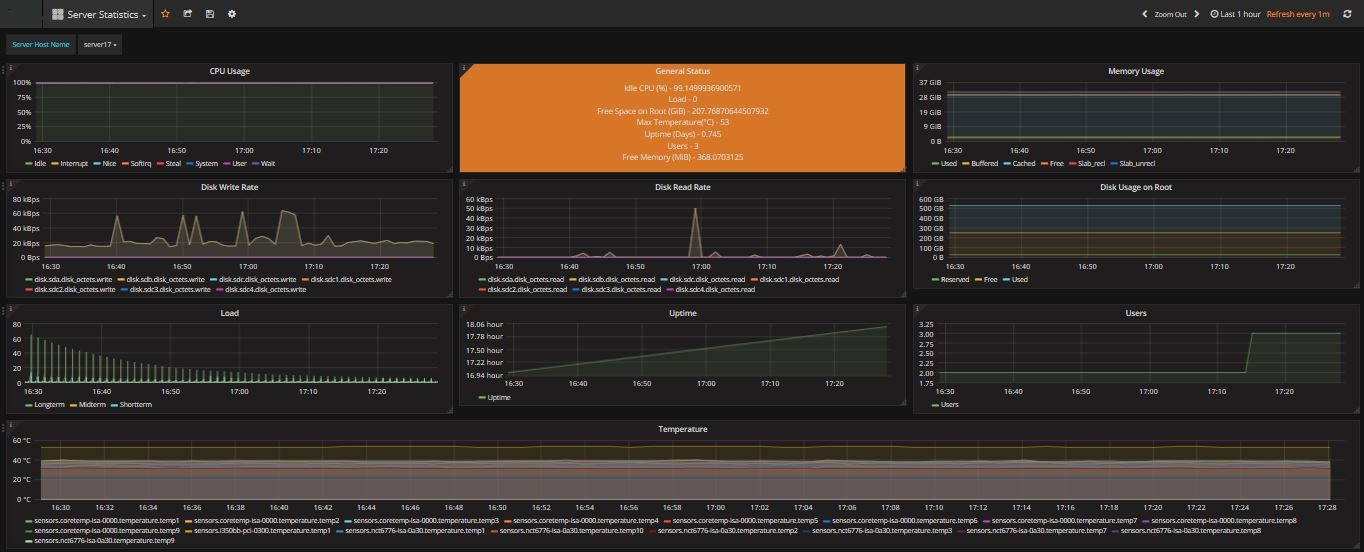
图106: 可用的LDLM Callback请求缓冲区数目面板



## 服务器仪表盘

服务器仪表盘 (图107) 显示了服务器的统计信息。

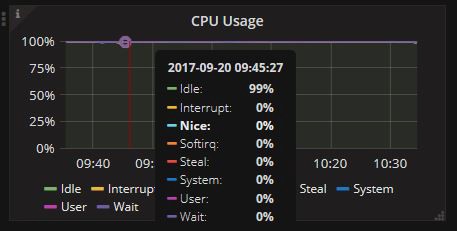
图107：服务器仪表盘



以下是在服务器仪表盘中可以找到的面板：

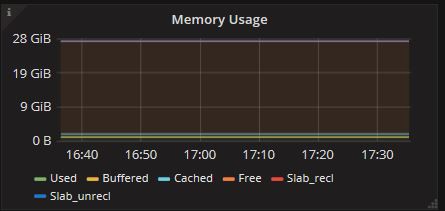
* **CPU使用率**面板（图108）显示了CPU在不同状态下的使用率，即CPU在执行用户代码、执行系统代码、等待数据读写操作、空闲时所占用的时间。

图108：CPU 使用率面板



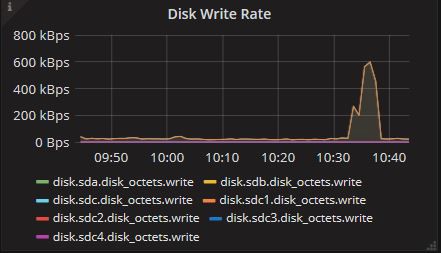
* **内存使用率**面板（图109）显示了内存的使用情况，这些值来自于操作系统，包括：Used, Buffered, Cached, Free, Slab\_recl, Slab\_unrecl。

图109：内存使用率面板



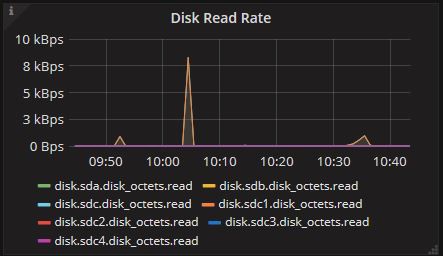
* 磁盘写速率面板（图110）显示出服务器磁盘写入速率随时间的变化。

图110：磁盘写速率面板



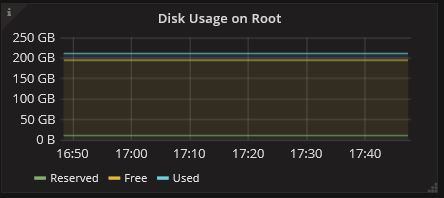
* **磁盘读速率**面板（图111）显示出服务器磁盘写入速率随时间的变化。

图111：磁盘读速率面板



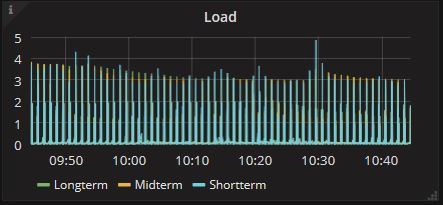
* **根目录磁盘使用率**面板（图112）显示了根目录的空闲空间、使用空间、预留空间。

图112：根目录磁盘使用率



* **负载**面板（图113）显示了服务器负载情况，即服务器运行队列中可运行的任务数量。相关信息由操作系统提供，分为三类：
  + **Shortterm（短期）**— 一分钟内的平均负载。
  + **Midterm（中期）**—五分钟内的平均负载。
  + **Longterm（长期）**—十五分钟内的平均负载。

图113：负载面板



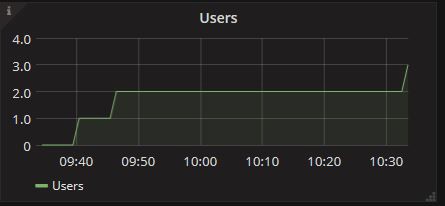
* 启动时间面板（图114）显示出服务器已经启动了多长时间。提供诸如平均运行时间或在一段时间内达到的最大正常运行时间等信息。

图114：启动时间面板



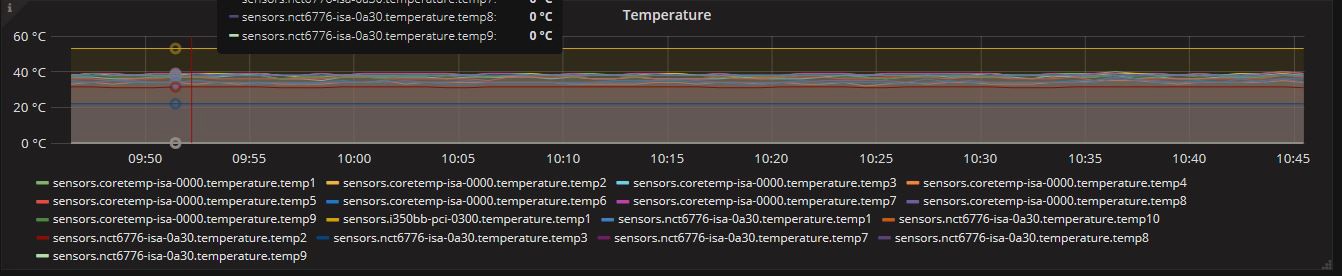
* **用户数**面板（图115）显示出登陆系统的用户数量。

图115：用户数面板



* **温度**面板（图116）显示出从各温度传感器中收集的温度信息。

图116：温度面板



## SFA 物理磁盘仪表盘

**SFA物理磁盘仪**表盘**（**图117）显示了SFA物理磁盘的相关信息。

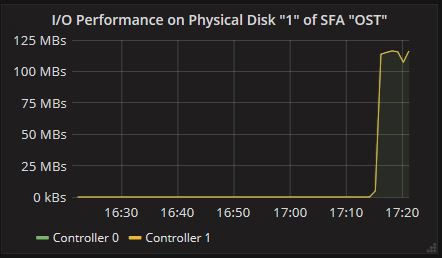
图117: SFA物理磁盘仪表盘



下面是可以在**SFA物理磁盘**表盘中可以找到的面板：

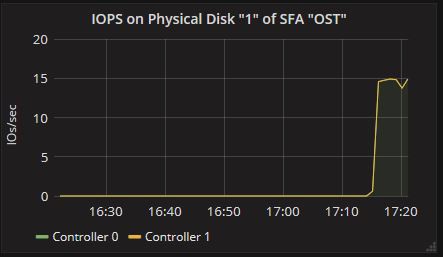
* **I/O 吞吐率**面板（图118）显示出该物理磁盘的I/O吞吐率。

图118：I/O 吞吐率面板



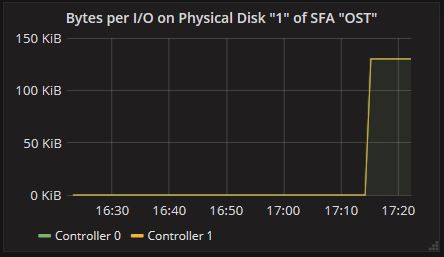
* **IOPS**面板（图119）显示出该物理磁盘每秒I/O操作数目。

图119：IOPS面板



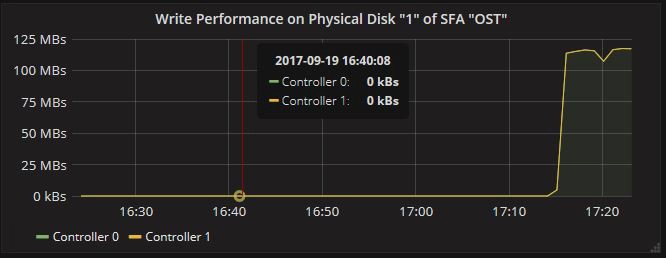
* **I/O大小**面板(图120) 显示出各控制器中该物理硬盘的每次I/O的平均大小。

图120：I/O 大小面板



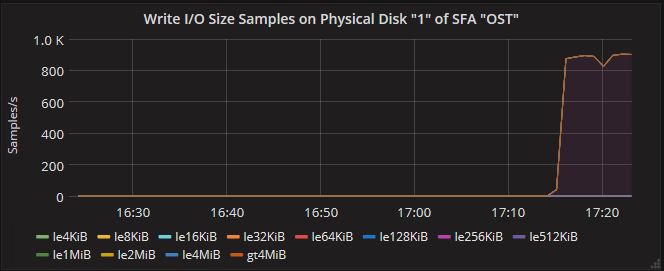
* **写数据吞吐率**面板（图121）显示出各控制器中，该物理硬盘的写入吞吐率。

图121：写数据吞吐率面板



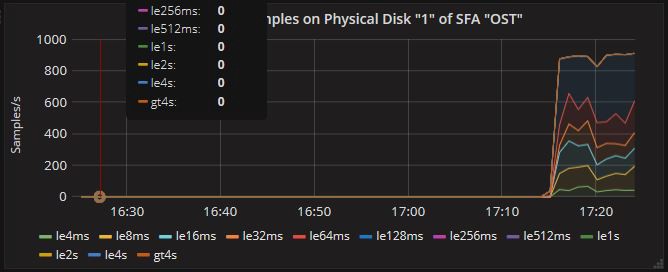
* **写数据的I/O大小样品采样数目**面板（图122）显示出该物理磁盘上每种I/O大小的采样数目。

图122：写数据的I/O大小样品采样数目面板



* **数据写入延迟样品采样数目**面板（图123）显示出显示出该物理磁盘上，每种I/O延迟的采样数目。

图123：数据写入延迟样品采样数目面板



## SFA虚拟磁盘表盘

**SFA虚拟磁盘**表盘（图124）显示出SFA虚拟磁盘表盘的信息。

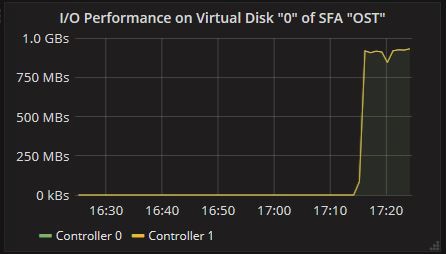
图124：SFA虚拟磁盘表盘



下面是可以在**SFA虚拟磁盘**表盘中找到的面板：

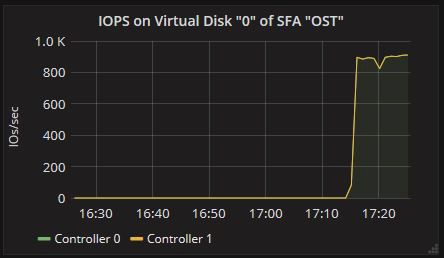
* I/O吞吐率面板（图125）显示出该虚拟磁盘上的I/O吞吐率。

图125：I/O吞吐率面板



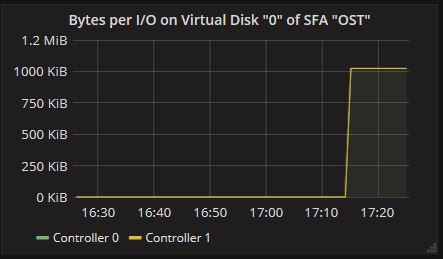
* IOPS面板（图126）显示出在不同的控制器上，该虚拟磁盘上的IOPS。

图126：IOPS面板



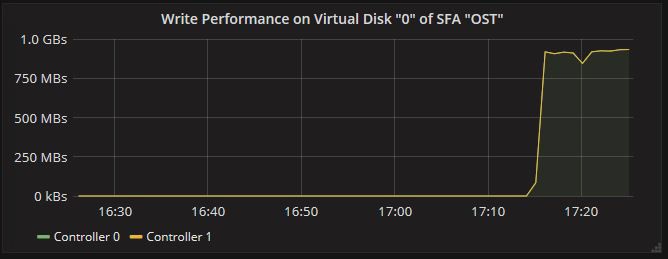
* I/O大小面板（图127）显示出各控制器上，该虚拟磁盘每次I/O的平均大小。

图127：虚拟磁盘每次I/O的字节数面板



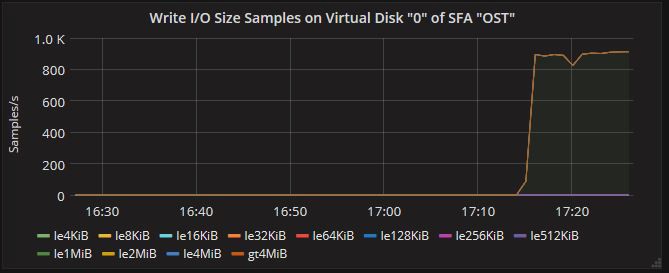
* 数据写入速率面板（图128）显示出每个控制器上，该虚拟磁盘的数据写入速率。

图128：虚拟磁盘的写入性能面板



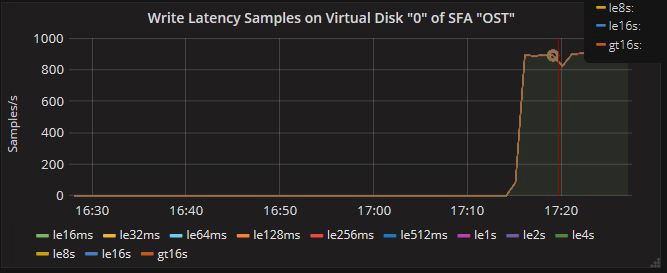
* 数据写入的I/O大小样品采集数目面板（图129）显示出该SFA虚拟磁盘上，数据写入的I/O大小的分布。

图129：数据写入的I/O大小样品采集数目面板



* 数据写入的延迟样品采集数目面板（图130）显示出该SFA虚拟磁盘上，数据写入延迟的分布。

图130：数据写入的延迟样品采集数目面板



# 压力测试

为了验证监控在高负载的集群环境下是否运行良好，我们设计了**collectd-stress2** 插件来进行压力测试。它是**Stress** 插件的升级版，它可以通过几个**collectd**客户端来模拟几百个服务器收集的数以万计的监控数据。

## 在代理节点安装插件

由于**stress2**插件会产生大量的模拟的监控数据，污染数据库，所以该插件默认不会在所有的客户端安装。当使用**esmon\_install** 部署完监控系统之后，用户可以选择某个客户端作为压测客户端。可以在ISO目录下找到collectd-stress2\*.rpm并通过以下命令进行安装：

rpm --ivh collectd-stress2\*.rpm

## 在代理节点更新配置文件

安装**stress2** RPM之后，更新配置文件/etc/collectd.conf, 添加下面的配置：

* **Thread**—定义了测试线程数目**。**
* **Metric**—定义了一个监控目标的所有属性,它可以指定多次，用来同时模拟不同的监控目标**，**它包含以下属性:
  + **Variable**—定义了监控目标变化的范围以及变化的速度，可以指定多次。
    - **Name**—定义了变量名。
    - **Number**—定义了变量变化的范围最大值。
    - **UpdateIterval**—定义了变量变更的间隔时间。
  + **Host**—定义了客户端的主机名，通常定义为"${key:hostname}"，由程序自动设置当前主机名，它和下面的**Plugin**，**PluginInstance**，**Type**，**TypeInstance**共同组成collectd 描述收集数据对象的区分符，详情请参见[Collectd 的描述符](https://collectd.org/wiki/index.php/Naming_schema)**。**
  + **Plugin**—定义了collectd 识别符中的plugin 成员**。**
  + **PluginInstance**—定义了collectd 识别符中的plugininstance 成员。
  + **Type**—定义了collectd 识别符中的type 成员,详细可参见[数据类型](https://collectd.org/wiki/index.php/Derive)**。**
  + **TypeInstance**—定义了collectd 识别符中的type instance成员。
  + **TsdbName**—定义了提交到数据库格式中的名字。
  + **TsdbTags**—定义了提交到数据库格式的标签，方便后期分类查找。

**下面是一个**/etc/collectd.conf**的例子。**

**例子:**

LoadPlugin stress2

<Plugin "stress2">

  Thread 32

  <Metric>

<Variable>

    Name "ost\_index"

    Number 10

    UpdateIterval 0

</Variable>

<Variable>

    Name "job\_id"

    Number 7000

    UpdateIterval 10

</Variable>

  Host "${key:hostname}"

  Plugin "stress-${variable:ost\_index:OST%04x}"

  PluginInstance "jobstat\_${variable:job\_id:job%d}"

  Type "derive"

  TypeInstance "sum\_read\_bytes"

  TsdbName "ost\_jobstats\_samples"

  TsdbTags "optype=sum\_read\_bytes fs\_name=stress ost\_index=${variable:ost\_index:OST%04x} job\_id=${variable:job\_id:job%d}"

   </Metric>

  <Metric>

<Variable>

    Name "mdt\_index"

    Number 10

    UpdateIterval 0

</Variable>

<Variable>

    Name "md\_stats"

    Number 10

    UpdateIterval 10

</Variable>

  Host "${key:hostname}"

  Plugin "stress-${variable:mdt\_index:MDT%04x}"

  PluginInstance "md\_stats"

  Type "derive"

  TypeInstance "open"

  TsdbName "md\_stats"

  TsdbTags "optype=open fs\_name=stress mdt\_index=${variable:mdt\_index:MDT%04x} mdt\_stats\_open=${variable:mdt\_stats\_open:%d}"

   </Metric>

</Plugin>

## 进行压力测试

在更改了配置文件之后，重启Collectd:

service collectd restart

通过查看/var/log/messages 日志，可以看到如下消息：

server11 collectd[20830]: stress2: time: 1.79244 for 70100 commits with 32 threads, 39108.70099 commits/second

上面信息显示**stress2** 插件已经成功加载运行，并且产生了大量的监控数据。

下面将以上面的配置文件为例，分析在如下指定的硬件环境下，如何分析对应的监控瓶颈。

* **操作系统版本：**CentOS7。
* **内存：**128GB。
* **处理器:** Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz。
* **硬盘：**Samsung SSD 850 2B6Q。

监控客户端和服务器端运行在同一台服务器上，Influxdb 数据存在SSD对应的Ext4文件系统上。

**前提条件：**

* **Collectd 收集数据间隔：**60秒。
* **Grafana 展示历史时间长度：**1小时。
* **Grafana 自动刷新间隔：**60 秒。
* **Collectd 运行时间：大于1个小时。**

**结论：**

* **Grafana 页面一直处于刷新状态：**监控过载。
* **Grafana 页面有空闲：**监控运行良好。

理论上Grafana 的刷新时间，等于数据库查询时间加上页面布局的加载时间。

我们可以通过查询数据库，来衡量数据库的查询性能，比如针对ESMON Grafana 默认的“Read Throughput per Job”查询，数据库端相应的查询语句：

influx -database esmon\_database –execute \

"SELECT "value" FROM "ost\_jobstats\_samples" WHERE ("optype" = 'sum\_read\_bytes' AND "fs\_name" = 'stress') AND time >= now() - 1h GROUP BY "job\_id""

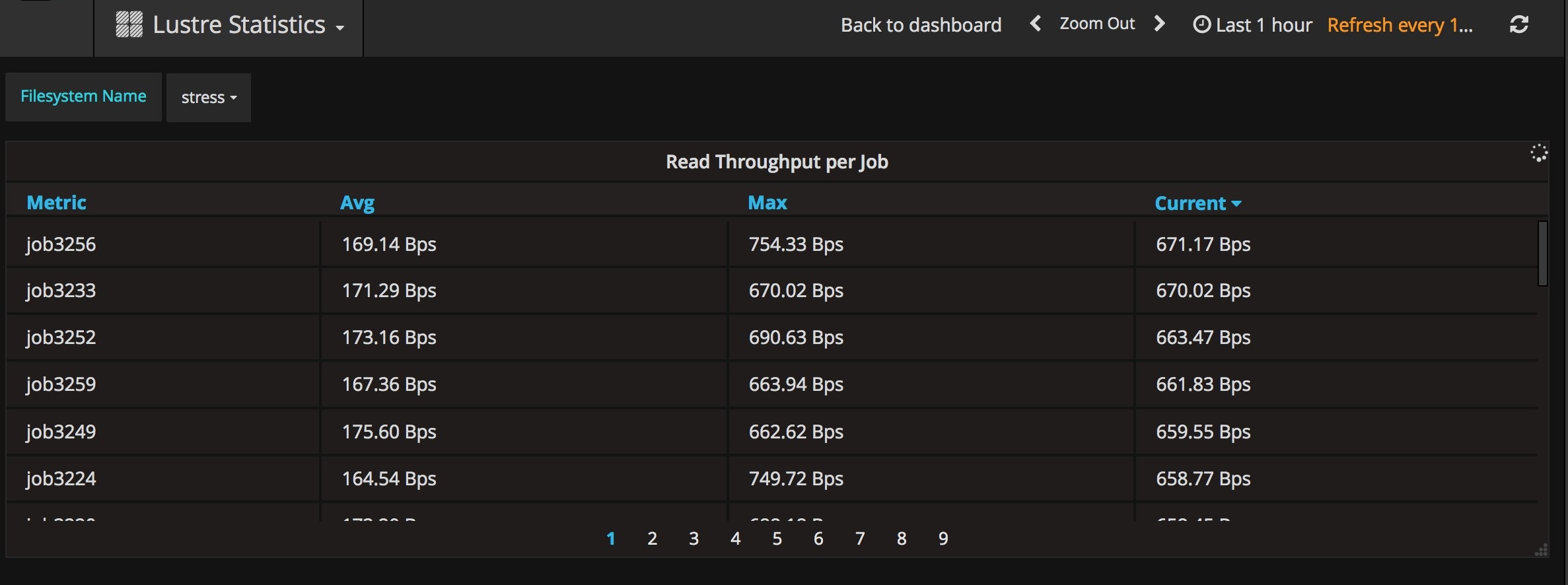
随着监控软件的运行，可以在数据库后端的host 上执行该条语句，验证数据库端查询的时间。

如图71所示，Influxdb 在一个小时以内，数据查询时间呈线性增长，因为查询获得的数据量会随着时间增长而增长。但是一个小时以后，查询时间稳定在二十多秒。

图131：Influxdb Query Time 变化曲线图

验证完了数据库端的负载，我们还需要验证Grafana界面的加载情况，如图72 所示，通过Grafana登陆查看“**每个任务读吞吐量**”。

图132：每个任务读吞吐压测面板



如果页面一直处于刷新状态，页面在60秒以内无法加载出来，则说明当前配置下监控处于过载状态，说明当前需要升级监控的硬件配置，或者增大收集时间间隔。否则则表明监控运行良好。通过不断调整*/etc/collectd.conf中的***job\_id**的Number，观察页面加载延迟，可以得到当前硬件配置下可支持的最大压力。测试表明，该设置条件下，监控可以支持10 个OST上并行运行的7000个作业。

# 故障排查

部署服务器收集了所有可用于调试的日志，并将其保存至的*/var/log/esmon\_install/[installing\_date]*目录下。一旦操作失败，相关错误信息将被输出至*/var/log/esmon\_install/[installing\_date]/error.log*文件中。通常，第一条错误信息包含了导致该操作失败的原因。