Implantação de um cluster de alto desempenho com OpenHPC: um relato prático na MackCloud

Allan Gustavo Fernandes[1], Calebe de Paula Bianchini[1]

[1]Faculdade de computação de informática – Universidade Presbiteriana Mackenzie
01302-907 – São Paulo – SP – Brasil

allan.fernandes@mackenzista.com.br, calebe.bianchini@mackenzie.br

Abstract. *This work presents the implementation of the OpenHPC project in a MackCloud cluster. The OpenHPC provides a complete ecosystem for the high-performance computing environment, providing rapid deployment and flexibility in management and usability. Currently, these are the main challenges, as each day these environments have become more complex, with varied applicability and with an increasing number of computational nodes.*

Resumo. *Este trabalho apresenta a implementação do projeto OpenHPC em um cluster na MackCloud. O OpenHPC fornece um ecossistema completo para o ambiente de computação de alto desempenho, provendo uma rápida implementação e flexibilidade no gerenciamento e usabilidade. Atualmente estes são os principais desafios, pois a cada dia estes ambientes têm se tornado mais complexos, com aplicabilidade variada e com um número cada vez maior de nós computacionais.*

1. Introdução

High-Performance Computing, também chamada de HPC, surgiu com objetivo de aumentar o poder computacional, em especial a capacidade de processamento. Uma das propostas foi um conjunto de computadores interligados para realização de processos em paralelo que teve início em meados dos anos 1960 pela IBM (BELL, 2014).

A popularização ocorreu por volta dos anos 1980 quando os microprocessadores de alto desempenho, redes de alta velocidade e ferramentas padronizadas para a computação distribuída ganharam força, tornando o HPC, uma necessidade crescente entre o meio acadêmico e comercial, a construção de conglomerados de computadores

ou simplesmente *cluster*, era uma solução de menor custo comparado aos supercomputadores da época (BUYYA ,1999).

No final dos anos de 1993, Donald Becker e Thomas Sterling criaram o projeto Beowulf no Centro de Excelência em Dados Espaciais e Ciências da Informação da *University Space Research Association* que tinha como inovação a utilização de duas redes distintas. A primeira para comunicação de processos e a segunda para as demais trocas de dados entre os computadores, essa inovação trouxe um aumento de desempenho e se popularizou no meio acadêmico. Sendo o modelo adotado por outros meios de pesquisa, inclusive a NASA, o que ajudou na disseminação do uso dos clusters. (Beowoulf Project History, 2007)

Os ambientes HPC por muitos anos obteve um crescimento massivo no desempenho, sendo o principal motivador a grande demanda gerada pelas comunidades cientificas, engenharia e indústria de manufatura. Apesar do alto custo de investimento em HPC, para cada dólar investido se obtém um retorno médio de $44 dólares, seja em retorno direto ou retorno indireto gerado por algum tipo de economia. (ROI, 2020).

Visando criar um ambiente mais flexível que permite aos usuários realizarem o maior número possível de pesquisas nas mais diversas áreas de conhecimento de forma eficiente, este trabalho tem como objetivo implementar um cluster de computadores utilizando OpenHPC. Com isso, esperamos diminuir o tempo de construção de um ambiente originalmente complexo devido à variedade de softwares, bibliotecas, compiladores (Cluster Computing with OpenHPC, 2016).

2. O projeto OpenHPC

Em 2015, durante a “*International Supercomputing Conference*”, foi realizado um painel de discussão intitulado de “*Community Supported HPC Repository & Management Framework*”, onde foi discutido sobre os componentes de softwares comumente necessários para construir *clusters* de computação. Após a positiva resposta da comunidade nasceu o projeto intitulado de OpenHPC, sendo mantido juntamente com a *Linux Foundation.* (What is OpenHPC, 2017)

O OpenHPC fornece uma coleção integrada e testada de componentes de software que junto com suporte a uma distribuição Linux padrão pode ser usada para implementar um *cluster* de computação completo. Os componentes abrangem todo o ecossistema de software HPC, incluindo ferramentas de provisionamento e administração de sistema, gerenciamento de recursos, serviços de E/S, além de ferramentas de desenvolvimento, bibliotecas numéricas e ferramentas de análise de desempenho (OpenHPC, 2020).

A arquitetura do OpenHPC é intencionalmente modular para permitir que os usuários finais selecionem os componentes fornecidos e para promover novas ideias em uma comunidade de contribuição aberta. O projeto fornece receitas para a construção de clusters usando os sistemas operacionais Linux CentOS e openSuSE em arquitetura x86_64, bem como arquiteturas arch64 (OpenHPC, 2020).

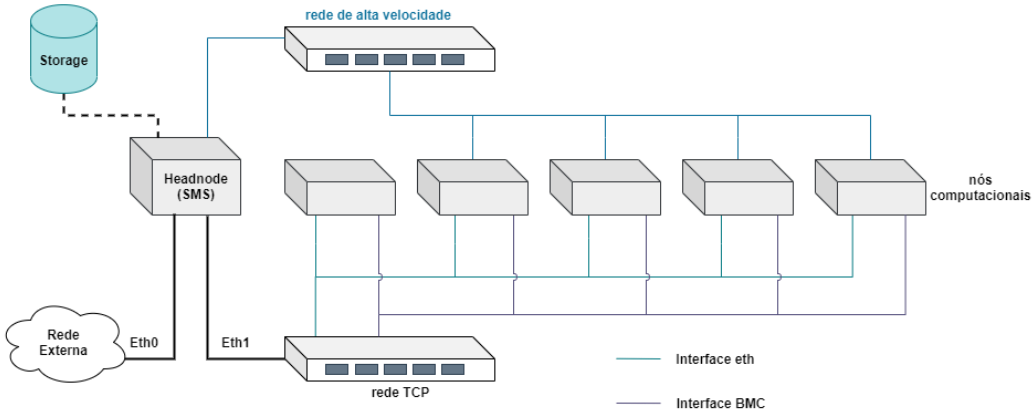
3. Implementação de cluster de computação

A implementação do cluster descrito neste artigo foi realizada utilizando a documentação do OpenHPC v1.3.9[1] para o sistema operacional CentOS 7.7 e Slurm 18.08. Além disso, scripts adicionais foram desenvolvidos em *shell script*.

3.1. Definições do ambiente e especificações técnicas

A disposição de rede física foi implementada como na figura abaixo.

Figura 1. Esquema de arquitetura de rede



Fonte: Desenhada pelo autor.

[1] Disponível em: <https://bit.ly/3pfHPb3>. Acesso em: 30 de maio de 2021

As definições das interfaces de rede foram: Rede Externa, Rede Interna, Rede InfiniBand, Rede BMC.

Para a Rede Externa, o endereço IP foi provido pela Gerência de Tecnologia e Inovação (GERTI) da Universidade Presbiteriana Mackenzie.

Para a Rede Interna, foi definido a faixa de IP 10.141.0.10/24.

Para a Rede InfiniBand (IPoB), foi definido a faixa de IP 10.149.0.10/24

Para a Rede BMC, foi definido a faixa de IP 10.148.255.0/24

O cluster foi criado no conjunto de máquinas as seguintes especificações:

Tabela 1: Especificações técnicas

	Quantidade	Fabricante	Modelo	Processador	Memória	Disco	GPU
Headnode	1	SuperMicro	6018R-MTR	2x Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz	4x 16GB DDR4 2400MHz	2x1TB	-
Compute Nodes	9	SuperMicro	6018R-MTR	2x Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz	4x 16GB DDR4 2400MHz	2x1TB	-
Graphic Node	1	SuperMicro	1028GR-TR	2x Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz	8x 16GB DDR4 2400MHz	2x1TB	NVIDIA K80 12GB

	Fabricante	Modelo	Capacidade
Storage	SuperMicro	6019P-ACR12L-OTO-75	120TB - SAS 3 (12 x10TB)

	Fabricante	Modelo
Rede InfiBand	Mellanox	Msx6012f,12 portas QSFP de 40Gb/s

Fonte: MackCloud

Para a configuração de rede de gerenciamento de *hardware* foi utilizada a *Intelligent Platform Management Interface* (IPMI), é uma interface padronizada para gestão e monitoramento de *hardware*, este padrão é mantido pelas empresas; Intel, Dell, HP e NEC. O IPMI utiliza o sistema *Baseboard Management Controller* (BMC), sendo este o responsável pela comunicação entre o sistema operacional e *hardware*.

O OpenHPC utilizará as funções da IPMI para implementar, gerenciar e monitorar os nós computacionais.

As definições de variáveis e nomenclaturas do projeto OpenHPC permite que o administrador realize a implementação que melhor atende as necessidades de seu

ambiente computacional, para tal desenvolvimento será definido uma série de variáveis e nomenclaturas.

Servidor SMS é a referência ao nó de gerenciamento do cluster, também conhecido como *Headnode, Master, Master node, Admin ou Admin Node.*

Tabela 2: Variáveis e nomenclaturas utilizadas.

Variável	Descrição
${sms_name}	Nome para o servidor SMS
${sms_ip}	Endereço IP interno do servidor SMS
${domain_name}	Domínio da rede interna
${sms_eth_internal}	Interface de rede interna do servidor SMS
${internal_netmask}	Máscara de rede da rede interna
${ntp_server}	Servidor local para sincronização de hora
${bmc_username}	Usuário do BMC para uso de IPMI
${bmc_password}	Senha do BMC para uso de IPMI
${num_computes}	Número total de nós computacionais
${c_ip[0]}, ${c_ip[1]}, ...	Endereço IP desejado para o nó
${c_bmc[0]}, ${c_bmc[1]}, ...	Endereço IP do BMC do nó
${c_mac[0]}, ${c_mac[1]}, ...	Endereço físico do nó
${compute_regex}	Expressão regular para combinar todos os nós
${compute_prefix}	Prefixo para o nome dos nós
${iso_path}	Caminho da ISO para a instalação via xCAT
${ohpc_repo_dir}	Caminho do repositório espelho do OpenHPC
${epel_repo_dir}	Caminho do repositório espelho do EPEL
${sysmgmtd_host}	BeeGFS Sistema de gerenciamento de hostname
${mgs_fs_name}	Lustre MGS nome de montagem
${sms_ipoib}	IPoB endereço para servidor SMS
${ipoib_netmask}	IPoB máscara de rede
${c_ipoib[0]}, ${c_ipoib[1]}, ...	IPoB endereço para o nó
${nagios_web_password}	Senha para acesso web do Nagios

Fonte: Manual OpenHPC (página, 7)

Todos os comandos foram executados como super usuário, ou seja, utilizando o usuário *root*.

As configurações dos nós computacionais também são executadas através do *Headnode* e são submetidas aos nós utilizando o comando *psh*[2] (*Parallel Remote Shell*),

[2] Disponível em: <https://xcat-docs.readthedocs.io/en/stable/guides/admin-guides/references/man1/psh.1.html>. Acesso em: 20 de março de 2021

este permite que sejam enviados uma série de comandos em paralelo para todos os nós elegidos.

Para a definição das variáveis que serão utilizadas durante a implementação do OpenHPC, foi desenvolvido um script[3], onde este realiza a leitura de um arquivo de texto contendo a lista de endereços físicos dos nós ordenados, onde a primeira linha do arquivo representa o primeiro nó, a segunda linha o segundo nós e assim sucessivamente.

A instalação do sistema operacional foi realizada no nó designado como *Headnode.*

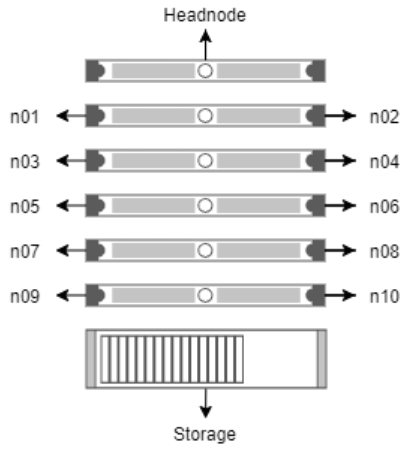
Na implementação do cluster em questão, foi desabilitado o *SELinux, Security-Enhanced Linux*, que permite aos administradores criarem políticas de acesso aos recursos do sistema operacional e/ou hardware.

O arquivo *hosts* deve ser atualizado no *Headnode* para que contenha o endereço IP e o nome de máquina designado ao mesmo.

3.2. Provisionamento

A identificação prévia dos nós foi realizada coletando os endereços físicos, para que no momento do provisionamento do sistema operacional mantivéssemos a visão desejada do *rack*, como a exibida na imagem abaixo.

Figura 2: Visão frontal do rack com identificação dos nós



Fonte: Desenhada pelo autor.

[3] Disponível em: <https://bit.ly/3c4EJ47>. Acesso em: 30 de maio de 2021.

Instalação do xCAT[4] e provisionamento do nós foi realizado utilizando o sistema operacional CentOS 7.7 na modalidade *Stateful*, ou seja, o sistema operacional será instalado no próprio disco rígido do nó computacional.

O xCAT (*Extreme Cloud Administration Kit*), é um software de código fonte aberto desenvolvido pela IBM para a automatização do provisionamento de sistemas operacionais.

Uma vez que o repositório do xCAT foi adicionado aos repositórios do *Headnode* foi necessário realizar a inclusão dos pacotes ao sistema operacional, realizar o carregamento do ambiente e registrar a interface de rede de provisionamento.

A construção das imagens base para provisionamento foi realizado utilizando o comando *copycds*, para isso é necessário que a imagem ISO tenha sido salva previamente no diretório designado na variável *${iso_path}*.

A inclusão dos nós computacionais na base de dados do xCAT foi realizada criando um grupo nomeado de *compute*, este agrupamento facilitará o gerenciamento do *cluster* e a execução de comandos em paralelo pelo *psh.*

O provisionamento dos nós computacionais é realizado via rede utilizando o ambiente de pré-execução (PXE, *Preboot Execution Environment*), este ambiente permite que os nós computacionais iniciem a instalação do sistema operacional designado e através dos scripts do xCAT realize a instalação de forma automatizada.

Uma vez que o *Headnode* e os nós computacionais estão com os sistemas operacionais devidamente instalados e configurados, os componentes dos OpenHPC são adicionados.

Criou-se um repositório local, pois assim os nós computacionais poderão encontrar os pacotes diretamente no *Headnode* através da rede local.

[4] Disponível em: <https://xcat-docs.readthedocs.io>. Acesso em: 20 de março de 2021

O ambiente HPC necessita que o relógio dos nós estejam sincronizados, a sincronização é realizada utilizando o relógio do *Headnode* através do serviço NTP (*Network Time Protocol*).

3.3. Configuração do Slurm e rede Infiniband

O Slurm, será o gerenciador de carga de trabalhos (*workload manager)* e o agendador de tarefas (*job scheduler*), este será responsável por gerir os recursos computacionais disponível no cluster, juntamente com as requisições de processamento, estas requisições são comumente chamadas de *job* e estes são submetidos a uma fila de execução, na qual aguardará até o momento em que os recursos solicitados estejam disponíveis.

A Configuração da rede de alta velocidade utilizada é a rede InfiniBand, para o cluster em questão, é possível atingir até 40 Gigabits por segundo (40Gb/s).

Os componentes OpenHPC adicionados aos nós computacionais são:

- Repositório EPEL
- Repositório local do OpenHPC
- Pacotes de cliente para o Slurm (*workload manager*)
- Pacotes de suporte a rede InfiniBand

O repositório EPEL (*Extra Packages for Entreprise Linux*), é uma seleção de pacotes que incluem módulos para *Python, Perl, Ruby gems*, entre outros pacotes extras para linguagens de programação.

No caso utilizaremos o repositório EPEL para estender as funcionalidades da linguagem Perl e Python.

O repositório OpenHPC é montado localmente no *Headnode* e os nós computacionais o acessam via rede interna, melhorando o desempenho durante o provisionamento dos nós computacionais. Este é o principal repositório, pois todas as funcionalidades do cluster são implementadas através da inclusão dos seus pacotes.

Os pacotes para o cliente Slurm e de suporte a rede InfiniBand são instalados e inclusos nos nós computacionais, sendo sua configuração facilitada pela integração do projeto OpenHPC com o sistema operacional.

O ambiente foi customizado permitindo que os *jobs* utilizem a rede InfiniBand para comunicação durante a execução de processos paralelos e para o tráfego de arquivos necessários à sua execução, em uma área destinada para tal sendo esta chamada de área de *scratch*.

Para que seja possível utilizar toda a capacidade da rede InfiniBand é necessário configurar o sistema operacional para que não realize o travamento de memória em todo o ambiente do *cluster*.

3.4. Sistemas de monitoramento

Para o monitoramento do ambiente do *cluster* utilizaremos três ferramentas, o Nagios[5], o Ganglia e o NHC[6], juntos estas ferramentas trazem um conjunto completo de monitoramento, notificações e tomada de decisão.

O Nagios é uma ferramenta de código fonte aberto para o monitoramento de sistemas e infraestrutura, sendo integrado ao projeto OpenHPC, o que facilita sua implementação em todo o *cluster*.

Serve para monitorar os serviços e processos em execução em todo o *cluster*, bem como a fazer a comunicação de rede, indicando e gerando alertas sobre a saúde geral de todo o ambiente HPC.

Após a implementação do Nagios, o monitoramento do *cluster* será iniciado automaticamente, o acesso será através do *Headnode* via web *http://<hostname>/nagios* ou *http://<endereço_ip>/nagios*, o usuário padrão para acesso é o *nagiosadmin* e a senha a será a mesma definida na variável de ambiente *${nagios_web_password}.*

A seguir temos um exemplo da tela de monitoramento dos detalhes da situação dos serviços para todos os computadores.

---

[5] Dísponivel em: <https://www.nagios.org/documentation>. Acesso em: 20 de março de 2021

[6] Disponivel em: <https://lbnl-node-health-check.readthedocs.io/en/latest/README.html>. Acesso em: 25 de março de 2021

Figura 3. Detalhe dos status dos serviços dos hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	04-18-2021 15:26:51	1d 16h 31m 43s	1/4	OK - load average: 0.14, 0.13, 0.14
	Current Users	OK	04-18-2021 15:25:29	103d 23h 51m 2s	1/4	USERS OK - 1 users currently logged in
	PING	OK	04-18-2021 15:24:22	103d 23h 52m 56s	1/4	PING OK - Packet loss = 0%, RTA = 0.06 ms
	Root Partition	OK	04-18-2021 15:26:51	103d 23h 52m 17s	1/4	DISK OK - free space: / 786936 MB (90.21% inode=100%):
	SSH	OK	04-18-2021 15:23:48	103d 23h 55m 46s	1/4	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Swap Usage	OK	04-18-2021 15:24:00	103d 23h 51m 40s	1/4	SWAP OK - 100% free (32191 MB out of 32191 MB)
n01	CPU Load	OK	04-18-2021 15:20:10	1d 22h 28m 24s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:27:49	1d 22h 30m 45s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:19:28	1d 22h 31m 8s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:23:11	1d 22h 25m 24s	1/3	PROCS OK: 0 processes with STATE = Z
n02	CPU Load	OK	04-18-2021 15:27:43	1d 22h 30m 51s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:20:59	1d 22h 27m 35s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:27:59	1d 22h 30m 35s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:21:12	1d 22h 29m 22s	1/3	PROCS OK: 0 processes with STATE = Z
n03	CPU Load	OK	04-18-2021 15:18:40	1d 22h 29m 54s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:27:41	1d 22h 30m 54s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:19:28	1d 22h 29m 8s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:19:48	1d 22h 30m 49s	1/3	PROCS OK: 0 processes with STATE = Z
n04	CPU Load	OK	04-18-2021 15:27:20	1d 22h 21m 14s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:23:29	1d 22h 25m 5s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:18:40	1d 22h 29m 54s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:28:18	1d 22h 30m 16s	1/3	PROCS OK: 0 processes with STATE = Z
n05	CPU Load	OK	04-18-2021 15:19:58	1d 22h 30m 39s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:27:12	1d 22h 21m 22s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:20:39	1d 22h 27m 58s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:19:31	1d 22h 29m 3s	1/3	PROCS OK: 0 processes with STATE = Z
n06	CPU Load	OK	04-18-2021 15:18:28	1d 22h 30m 6s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:20:04	1d 22h 30m 30s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:18:48	1d 22h 29m 47s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:19:09	1d 22h 29m 25s	1/3	PROCS OK: 0 processes with STATE = Z
n07	CPU Load	OK	04-18-2021 15:19:58	1d 22h 28m 38s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:18:38	1d 22h 29m 56s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:20:14	1d 22h 30m 20s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:23:58	1d 22h 24m 36s	1/3	PROCS OK: 0 processes with STATE = Z
n08	CPU Load	OK	04-18-2021 15:19:11	1d 22h 29m 23s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:21:01	1d 22h 27m 33s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:18:48	1d 22h 29m 47s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:19:32	1d 22h 31m 2s	1/3	PROCS OK: 0 processes with STATE = Z
n09	CPU Load	OK	04-18-2021 15:24:08	1d 22h 24m 26s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:28:09	1d 22h 30m 25s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:20:49	1d 22h 27m 47s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:20:33	1d 22h 30m 1s	1/3	PROCS OK: 0 processes with STATE = Z
n10	CPU Load	OK	04-18-2021 15:26:47	1d 22h 31m 48s	1/3	OK - load average per CPU: 0.00, 0.00, 0.00
	Current Users	OK	04-18-2021 15:24:18	1d 22h 34m 17s	1/3	USERS OK - 0 users currently logged in
	SSH Monitoring	OK	04-18-2021 15:25:56	1d 22h 34m 40s	1/3	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Zombie Processes	OK	04-18-2021 15:25:08	1d 22h 33m 28s	1/3	PROCS OK: 0 processes with STATE = Z

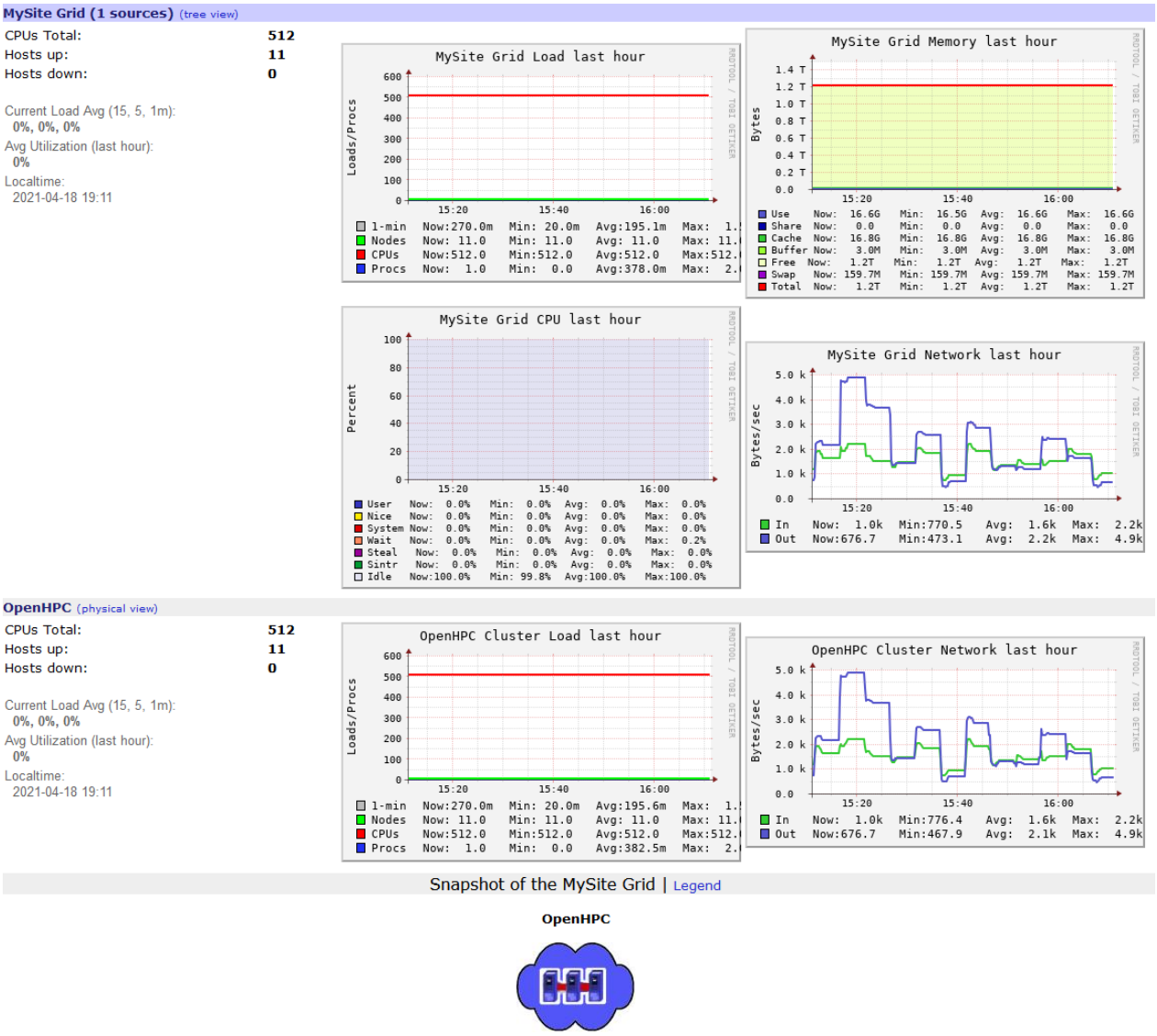
Fonte: Serviço de monitoramento do Nagios

O Ganglia é um sistema de monitoramento distribuído escalonável para HPC. Ele aproveita tecnologias amplamente utilizadas para transporte de dados compactos e portátil e está atualmente em uso em milhares de *clusters* em todo o mundo.

O serviço é provido através do acesso web, sendo possível acessá-lo através do *Headnode* via web, *http://<hostname>/ganglia* ou *http://<endereço_ip>/ganglia.*

O Ganglia permite aos usuários e aos administradores do *cluster* a visualização remota em tempo real ou o histórico estatístico do uso de recursos, como a figura abaixo.

Figura 4. Detalhe dos status o uso de recursos do cluster



Fonte: Serviço de monitoramento do Ganglia

O NHC é um serviço que verifica a saúde dos nós computacionais do *cluster* e é integrado ao Slurm. Este serviço faz uma verificação periódica na qual ao constatar que um nó está com problemas, alterará o seu estado para indisponível, fazendo com que o *cluster* tenha um aumento em sua confiabilidade, pois evitará que os *jobs* executem no nó computacional e tenha um estado de saída falho.

A integração do NHC é realizada adicionando as entradas diretamente no arquivo de configuração padrão do Slurm, a verificação de saúde foi habilitada para 300 segundos de intervalo entre as verificações.

3.5. Integração do ecossistema

O OpenHPC fornece uma variedade de pacotes que possuem uma fácil integração ou uma variante do mesmo pacote que já vem compilada a integração desejada, essas opões pode sem encontradas na documentação do OpenHPC.

O *munge*[7] é um serviço de autenticação para a criação e validação de credenciais. Ele é projetado para ser altamente escalonável para uso em um ambiente de *cluster* HPC. Permitindo que um processo autentique o UID (*User ID*, identificador único de usuário) e o GID (*Group ID*, identificador único de grupo) de outro processo local ou remoto dentro de um grupo de computadores.

O *mrsh* é um utilitário de *shell* remoto seguro, como o *ssh*, o *mrsh* usa o *munge* para autenticação e criptografia. Ao usar a instalação *munge* usada pelo Slurm o *mrsh* fornece acesso *shell* para o sistema usando a mesma chave *munge* sem ter que rastrear as chaves *ssh*. Assim como o *ssh*, o *mrsh* provê um comando de cópia remota, o *mrcp*.

Os componentes de desenvolvimento fornecem uma coleção de software que permite aos usuários desenvolverem, depurarem, compilarem e executarem seus próprios programas dentro do cluster.

O OpenHPC fornece em conjunto um ferramentas, o *toolchain*, que realiza a rápida configuração de ambientes de desenvolvimento e execução, permitindo que estes sejam facilmente modificados de acordo com a necessidade de cada programa ou usuário.

[7] Disponível em: <https://dun.github.io/munge/>. Acesso em: 30 de janeiro de 2021

O OpenHPC provê uma variedade de pacotes MPI tanto para o desenvolvimento quanto para a execução e para diversos tipos de transporte de rede.

Abaixo temos uma tabela comparativa com os pacotes MPI disponíveis e os respectivos suportes de transporte de rede.

Tabela 2: Variantes de pacotes MPI.

	Ethernet	InfiniBand	Omni-Path
MPICH	x		
MVAPICH		x	
MVAPICH (psm2)			x
OpenMPI	x	x	x
OpenMPI (PMIx)	x	x	x

Fonte: Manual OpenHPC (página, 22)

No cluster em questão foram instalados os pacotes OpenMPI, MPICH e MVAPICH.

A instalação dos pacotes de desenvolvimento MPI, disponibiliza de forma automática para os usuários todo o conjunto de ferramentas incluindo os compiladores básicos necessários com a pilha MPI.

Apesar do Slurm ser instalado em todo o *cluster*, sua configuração foi realizada somente no *Headnode*. Enquanto a configuração nos nós computacionais é realizada copiando o arquivo de configuração do Slurm e chave do *munge*.

A configuração se dará habilitando os serviços do *munge* e Slurm, posteriormente iniciando-os, na mesma ordem.

Após os serviços terem sido inicializados no *Headnode*, o mesmo processo será realizado, mas desta vez para inicialização nos nós computacionais.

3.6. Personalização

Foi realizada a inclusão de uma segunda fila de execução onde somente o nó computacional que possuí GPU foi incluído como recurso, para isso foi editado o arquivo de configuração do Slurm.

Comando 1. Inclusão de fila gpu

```
1    # echo “PartitionName=gpu Nodes=n10 Default=NO State=UP” >> /etc/slurm/slurm.conf
```

Fonte: Desenvolvido pelo autor

Os nós computacionais no Slurm podem ser verificados utilizando o comando *sinfo*.

3.7. Teste de desempenho da rede InfiniBand

O OpenHPC possui uma série de pacotes de código fonte aberto para auxiliar na análise de desempenho.

Foi realizado o teste de desempenho somente na interface de rede InfiniBand, sendo que vinha apresentando baixo desempenho nos testes com a implementação original do cluster.

O fabricante especifica uma taxa de transmissão de 40Gb/s para a rede InfiniBand sendo essa a rede de alta velocidade e baixa latência interna do *cluster em questão.*

Para realização dos testes, foi utilizado comando Linux “*qperf*”, onde se mensurou taxas médias de 37,83 Gb/s, 5% a menos que o esperado. Como podemos observar na tabela abaixo.

Tabela 2: Resultados obtidos em Gb/s

	Headnode	n01	n02	n03	n04	n05	n06	n07	n08	n09	n10
Headnode	-	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84
n01	37,68	-	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84
n02	37,60	37,84	-	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84
n03	37,84	37,84	37,84	-	37,84	37,84	37,84	37,84	37,84	37,76	37,84
n04	37,84	37,84	37,84	37,84	-	37,84	37,84	37,84	37,84	37,84	37,84
n05	37,68	37,84	37,84	37,84	37,84	-	37,84	37,84	37,84	37,84	37,84
n06	37,68	37,84	37,84	37,84	37,84	37,84	-	37,84	37,84	37,84	37,84
n07	37,68	37,84	37,84	37,84	37,84	37,84	37,84	-	37,84	37,84	37,84
n08	37,68	37,84	37,84	37,84	37,84	37,84	37,84	37,84	-	37,84	37,84
n09	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	-	37,84
n10	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	-
Média	37,74	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,84	37,83	37,84

Fonte: Mensurado pelo comando qperf.

4. Conclusão

Apesar de um grande tempo consumido para que fossem realizadas as configurações iniciais do ambiente e para o desenvolvimento dos scripts, foi possível realizar o

provisionamento dos nós computacionais em um tempo médio de 10 minutos, proporcionando uma rápida expansão e uma fácil manutenção do ambiente, o que é de grande valia.

Além disso, obtivemos uma taxa de comunicação performática e homogênea devido a boa integração entre as bibliotecas do OpenHPC e o sistema operacional.

Trabalhos futuros, será realizada a avalição de desempenho de todo ambiente, utilizando o pacote LinPack[8], este pacote é amplamente utilizado para a avaliação de desempenho de clusters, sendo válidos os dados coletados para a classificação no site top500.org, que exibe e classifica os mais poderosos clusters do mundo.

5. Bibliografia

BELL, Gordon, **Supercomputers: The Amazing Race** (A History of Supercomputing, 1960-2020), California, 2014

Beowoulf Project History, 2007. Disponível em: <http://www.beowulf.org/overview/history.html>. Acesso em: 14 de fevereiro de 2021.

BUYYA, R, **High Performance Cluster Computing: Architectures and Systems, Vol. 1**, Prentice Hall, 1999

HAGER, Georg, Wellein Gehard, **Introduction to High Performance Computing for Scientist and Engineers**, Boca Raton, Taylor & Francis, 2011

HWANG, Kai, FOX, Geoffrey C., DONGARRA, Jack J., **Distributed and Cloud Computing:** From Parallel Processing to the Internet of Things, Singapura, Elsivier, 2012

Manual OpenHPC, 2020. Disponível em: <https://bit.ly/3vCS8YX>. Acesso em 21 de fevereiro de 2021.

OpenHPC, 2021. Disponível em: < https://openhpc.community>. Acesso em: 11 de novembro de 2020.

[8] Disponível em: <https://www.netlib.org/benchmark/hpl/>. Acesso em: 09 de março 2021

ROI, 2020 Disponível em: https://www.hpcwire.com/2020/09/07/the-roi-on-hpc-44-in-profit-for-every-1-in-hpc. Acesso em 20 de março de 2021

SCHULZ, Karl W., BAIRD, C. Reese, BRAYFORD, David, GEORGIOU, Yiannis, KURTZER, Gregory M., SIMMEL, Derek, STERLING, Thomas, S UNDARARAJAN, Nirmala, VAN HENSBERGEN, Eric, **Cluster Computing with OpenHPC**, Indiana, Indiana University, 2016

What is OpenHPC, 2017. Disponível em: <https://opensource.com/article/17/11/openhpc>. Acesso em 21 de fevereiro de 2020.