

Sciflow-Bridge: Uma Ferramenta para Criação e Distribuição de Imagens de Contêineres para Workflows Científicos

Bruno da Silva Alves
Orientadora: Andreea S. Charão

04 de dezembro de 2019

Motivação

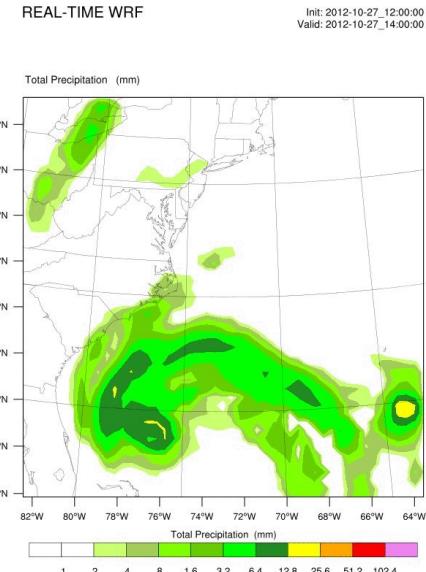
WRF(Skamarock W. et al., 2019): Sistema para previsão de tempo voltado tanto para pesquisa atmosférica quanto para aplicações de previsão.

UMA BARREIRA PARA A PESQUISA:

- A instalação de dependências é um processo demorado.
- Alguns pesquisadores não possuem conhecimentos técnicos.
- Reprodutibilidade é comprometida.
- A barreira inicial pode gerar desinteresse.

Passo da configuração	Nº de Comandos
Configuração de ambiente	20
Compilação das bibliotecas	47
Testes de compatibilidade	12
Compilação do WRF e WPS	23
Obtenção dos dados	10
Total	112

Fonte:
<http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation Tutorial.php>

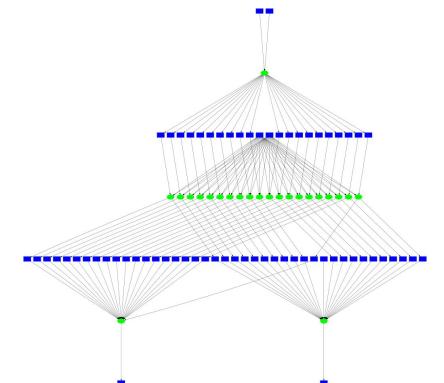


Workflows Científicos

- Workflows científicos permitem a descrição de experimentos constituídos por várias tarefas computacionais.
- O componente básico de workflows científicos são processos:

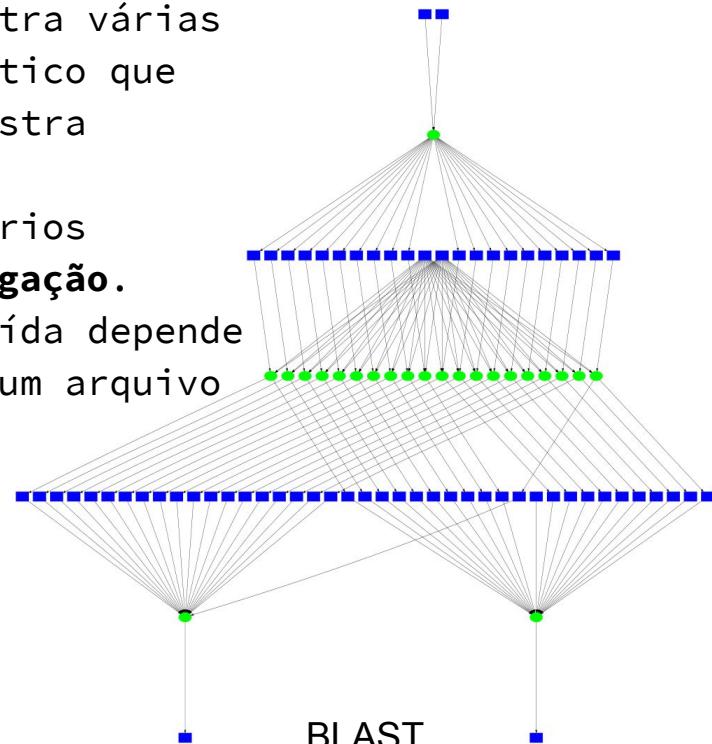
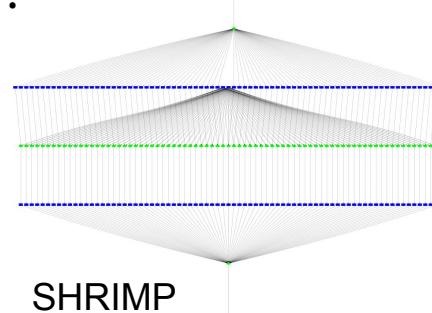
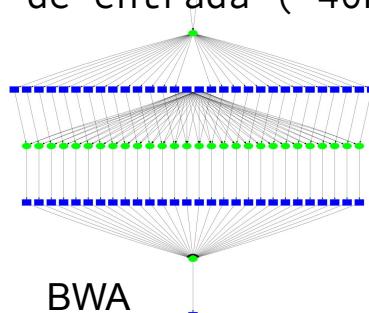


- A execução de workflows é orientada ao fluxo de dados.
- Sistemas de Workflow oferecem um ambiente para que os fluxos sejam modelados e executados.



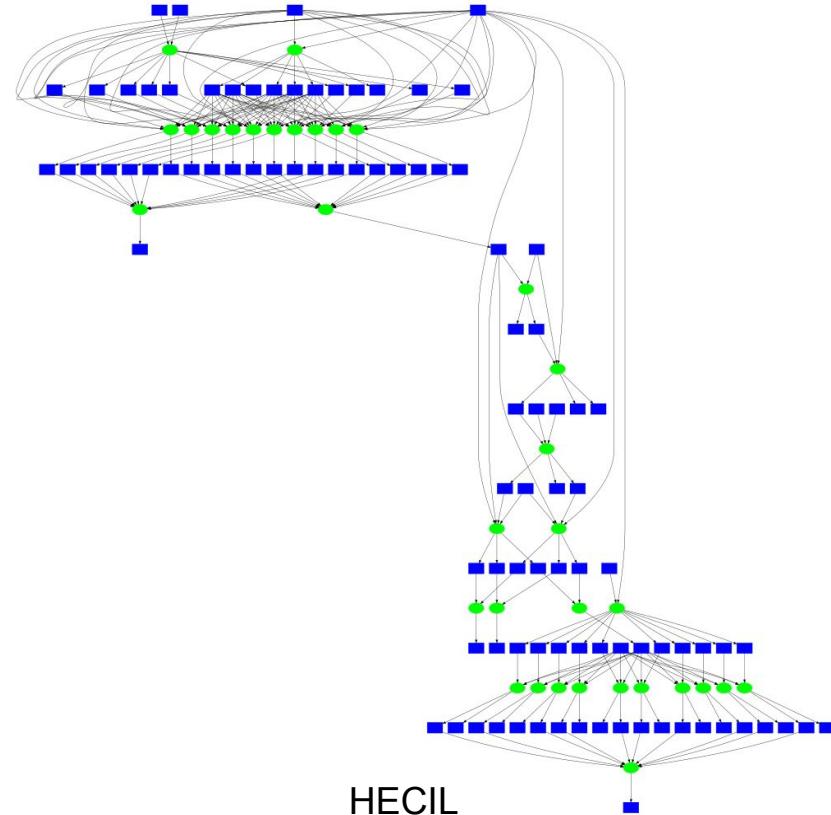
BLAST

- É um workflow da **bioinformática** que compara as semelhanças de uma proteína de referência contra várias outras sequências através de um método heurístico que retorna as correspondências semelhantes a amostra testada.
- Possui algumas estruturas básicas comuns à vários workflows, como: **distribuição, pipeline e agregação**.
- Cada tarefa que é executada de forma distribuída depende dos arquivos binários da pasta NT (565 MB) e um arquivo de entrada (~40KB).



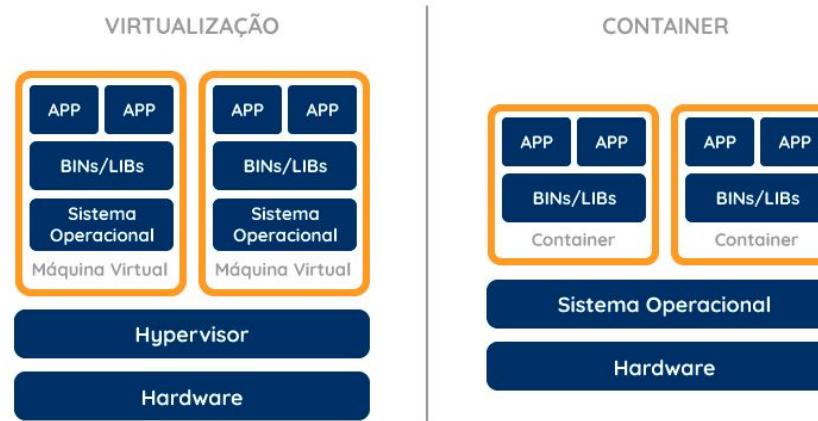
HECIL

- É uma aplicação que atua no sequenciamento de genomas, na correção de erros em leituras longas de sequenciamento.
- Possui dependências mais variadas do que o blast.
- Tarefas incluem scripts em python, execução de binários e códigos em Perl.



Contêineres

- É um tipo de virtualização em nível de SO.
- Utilizam algumas funcionalidades do Kernel, como Namespaces, CGroups, Chroot.
- Possuem níveis de performance próximos aos nativos.
- Acesso a disco mais rápido do que VMs.
- Possui questões de segurança e isolamento.



Fonte Imagem: <https://www.funcao.com.br/wp-content/uploads/2019/01/VirtualizacaoxContainer.png>

Docker e Docker Swarm

- Docker é uma das ferramentas mais populares para criação de contêineres.
- Docker:
 - Possui um repositório com mais de 2.000.000 de imagens de contêineres.
 - Suporte ativo dos desenvolvedores.
 - Possui uma comunidade de usuários ativos.
- Docker Swarm:
 - Orquestrador
 - Gerencia todo o ciclo de vida dos contêineres.

Pesquisa	Docker	LXC	OpenVZ
Repositórios no GitHub	415.415	1.181	427
Repositórios no BitBucket	8.864	42	11
Projetos no GitLab	Mais de 100	Mais de 100	7
Perguntas em que o nome da ferramenta aparecia no título	500	287	76
Perguntas com no mínimo uma resposta	500	211	63
Percentual de perguntas respondidas	100%	50%	36%

A ferramenta Sciflow-Bridge

- Visa integrar workflows científicos e contêineres.
 - Contêineres oferecem uma solução no provisionamento de ambientes.
 - Os contêineres podem encapsular todas as bibliotecas, executáveis e dependências de entrada de cada tarefa do workflow.
- A integração deve ser eficiente.
 - Contêineres não podem ser tratados simplesmente como um local onde todas as dependências são despejadas.
 - As dependências devem ser distribuídas entre os hosts.
 - As tarefas devem ser distribuídas entre os hosts.

Implementação

Analizar as dependências de cada tarefa.

Gerenciar as dependências de cada tarefa.

Transferir as dependências entre os nós do sistema.

Executar um workflow baseado em um modelo.

Definir uma estratégia para executar as tarefas do workflow.

Gerenciar as tarefas do workflow.

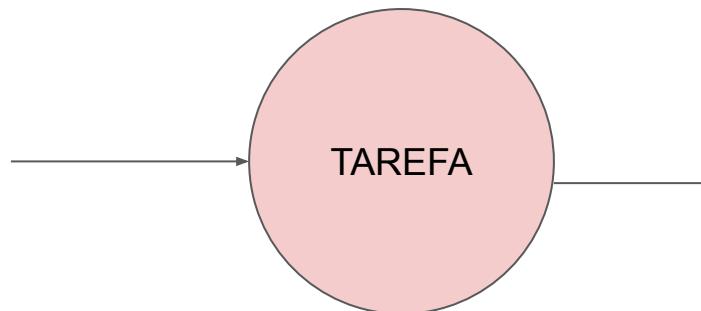
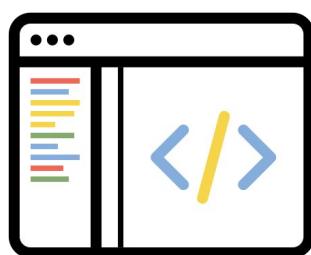
Gerenciar os contêineres.

Utilizar padrões existentes.



Análise de Dependências

- Existem dois tipos de dependências: dependências de entrada/saída e dependências implícitas.
- Para que a tarefa execute corretamente, é necessário prover um ambiente que contenha os dois tipos de dependências.

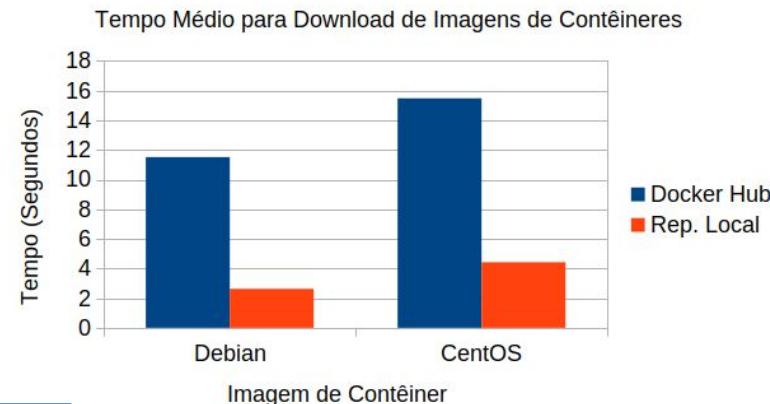
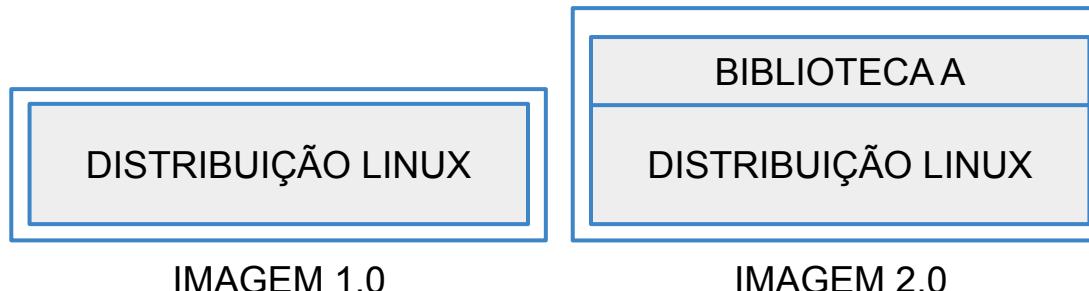


```
root@container: $tree dep
dep
|__ blastall
|__ etc
|   |__ ld.so.cache
|   |__ nsswitch.conf
|   \__ passwd
|__ lib
|   \__ x86_64-linux-gnu
|       |__ libc.so.6
|       |__ libm.so.6
|       |__ libnsl.so.1
|       |__ libnss_compat.so.2
|       |__ libnss_files.so.2
|       |__ libnss_nis.so.2
|       \__ libpthread.so.0
|__ nt
|   |__ nt.44.nhr
|   |__ nt.44.nin
|   |__ nt.44.nnd
|   |__ nt.44.nni
|   |__ nt.44.nsd
|   |__ nt.44.nsi
|   |__ nt.44.nsq
|   \__ nt.nal
\__ small.fasta.0

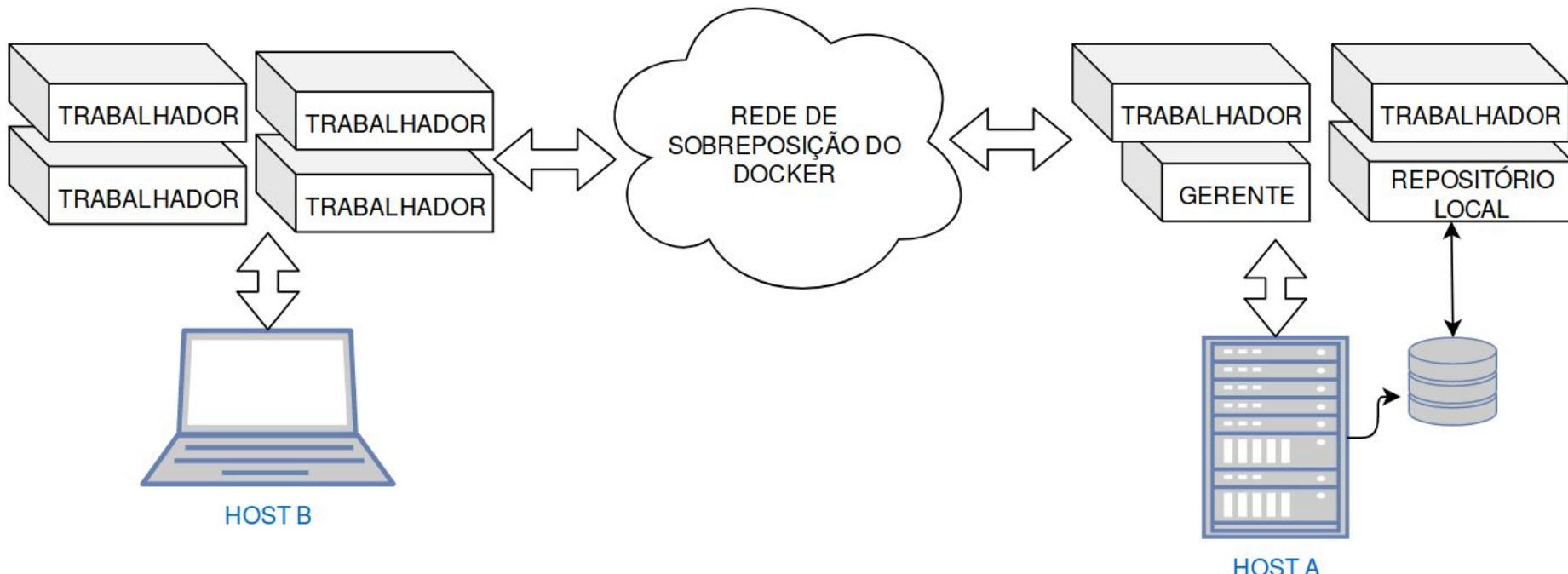
4 directories , 20 files
```

Transferência de Dependências

- Algumas opções para transferência de imagens de contêineres:
 - Tar file.
 - Dockerfile.
 - Via DockerHub.
- Solução adotada:
 - **Repositório Local de Imagens**
 - Envio de camadas.
 - Envio pela rede local.



Infraestrutura Proposta



Processador Intel(R) Core(TM) i5-3230M com 2 núcleos e 2 threads por core, 8 GB de memória RAM

Processador Intel(R) Xeon(R) com 4 núcleos e 2 threads por núcleo, 12 GB de memória RAM

Makeflow (ALBRECHT et al., 2012)

- Makeflow é um sistema de workflows científicos.
- Maneira simples para descrever workflows baseado na sintaxe de Makefiles.

```
saída_tarefa1 : dependencia_de_entrada1  
$./ tarefa1 -in dependencia_de_entrada1 -out saída_tarefa1  
  
saída_tarefa2 : saída_tarefa1  
$./ tarefa2 -in saída_tarefa1 -out saída_tarefa2
```

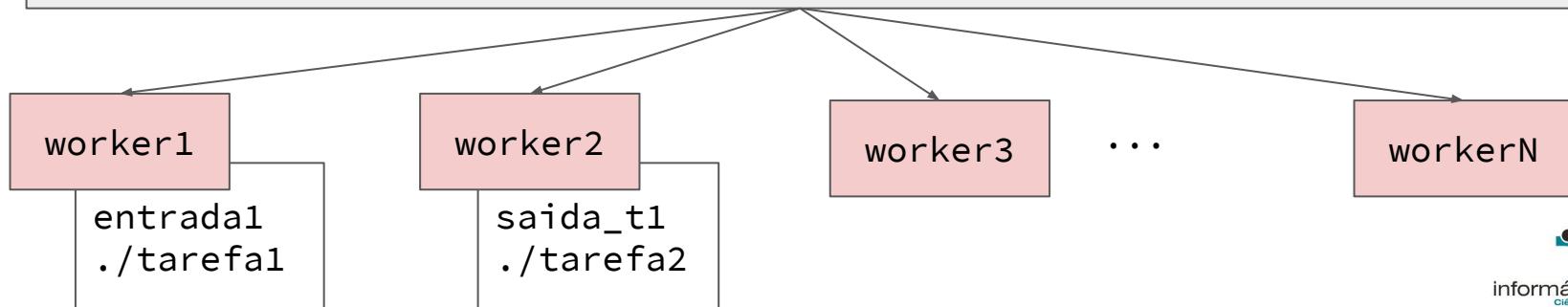
- Tarefa: Arquivos de entrada, arquivos de saída e comando para execução da tarefa.

Makeflow & Work Queue

- O Work Queue é o software responsável por distribuir as tarefas entre os nós trabalhadores.
- O Work Queue recebe as tarefas definidas pelo makeflow e as distribui.

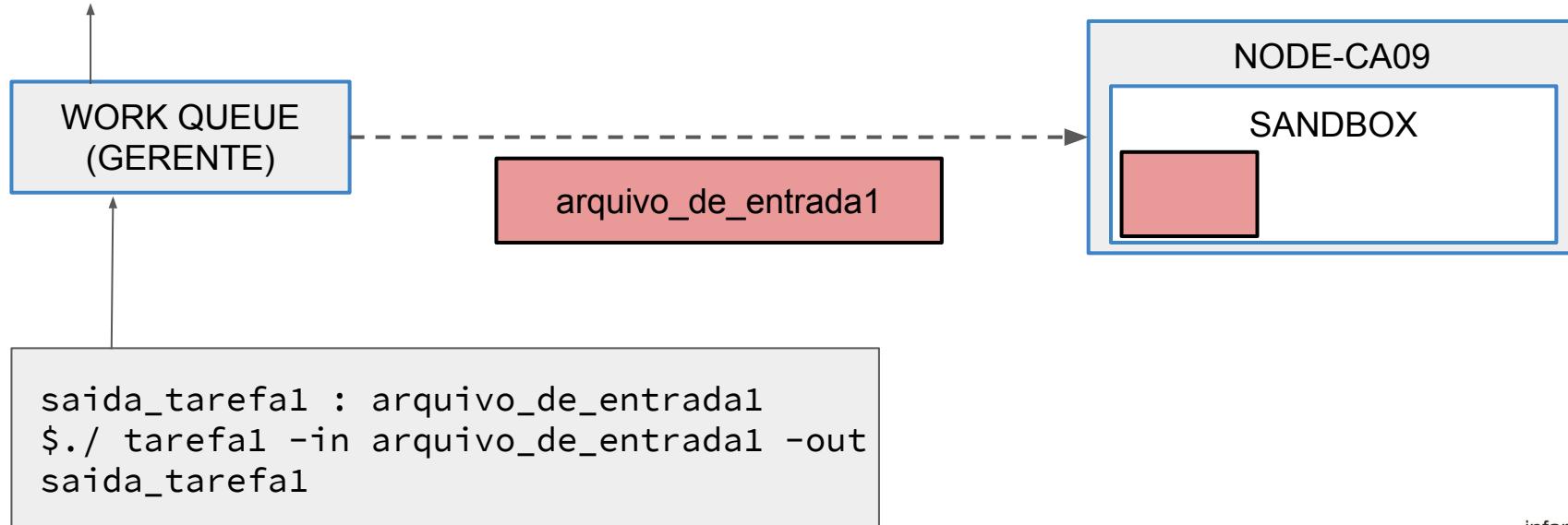
```
saida_tarefa1 : dependencia_de_entrada1  
$./ tarefa1 -in dependencia_de_entrada1 -out saida_tarefa1
```

```
saida_tarefa2 : saida_tarefa1  
$./ tarefa2 -in saida_tarefa1 -out saida_tarefa2
```



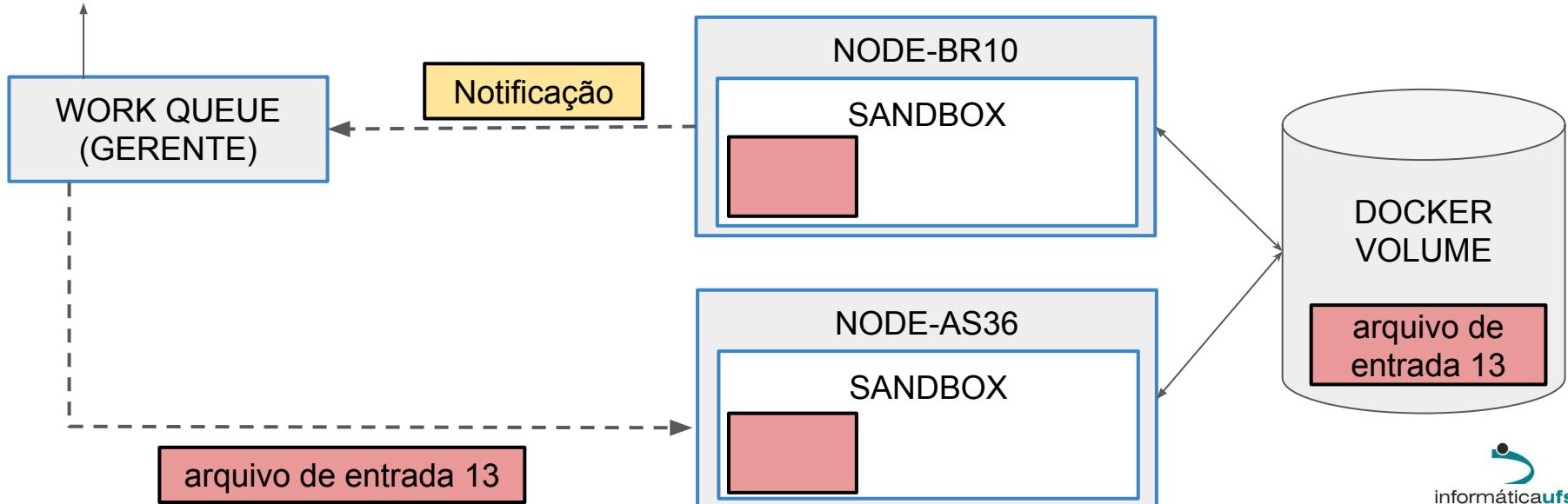
Work Queue - Uso de cache

Chave	Valor
Node-ca09	arquivo-entrada1-bb82, arquivo-de-entrada1-kj3n
Node-br10	arquivo-entrada13-ll08
Node-as36	arquivo-entrada13-ll08



Work Queue – Modificação

Chave	Valor
Node-ca09	arquivo-entrada1-bb82, arquivo-de-entrada1-kj3n
Node-br10	arquivo-entrada13-ll08
Node-as36	arquivo-entrada13-ll08



Resumo da Ferramenta

- A ferramenta Sciflow-Bridge é composta por um **conjunto de ferramentas** que tratam vários aspectos da integração de contêineres à Workflows Científicos.
- Pré-requisitos e soluções:
 - Análise de dependências: Script em Python
 - Gerenciamento de dependências: Contêineres + Makeflow
 - Transferência de dependências: Work Queue + Rede de sobreposição
 - Execução de um Workflow com base em um modelo: Arquivo do Makeflow
 - Gerenciamento das tarefas: Work Queue
 - Gerenciamento de contêineres: Docker Swarm + Docker Compose
 - Definição de uma estratégia: ?

Experimentos

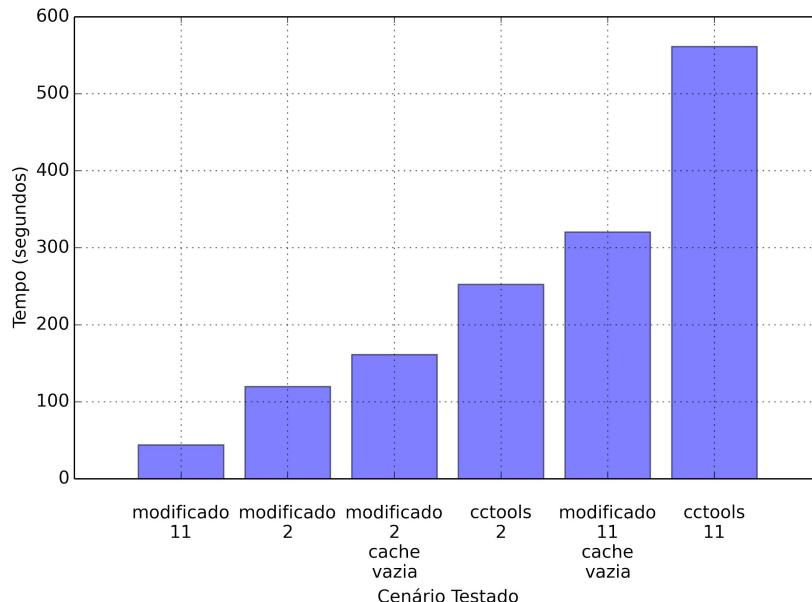
- Os experimentos realizados visam responder às seguintes questões:
 - É possível executar workflows com a ferramenta proposta?
 - Qual a melhor abordagem para posicionamento dos trabalhadores?
 - Deve-se utilizar um contêiner trabalhador para cada host?
 - Deve-se utilizar vários contêineres (1/thread)?
 - Qual a influência do workflow na escolha da abordagem?

Experimentos e Resultados

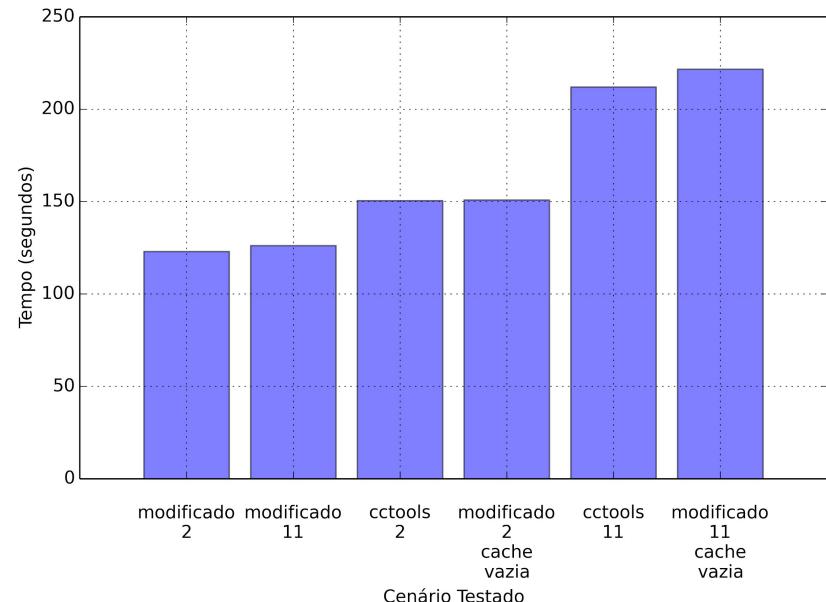
- Seis cenários foram propostos:
 - Makeflow + Work Queue com 11 trabalhadores - **1 worker/thread**
 - Makeflow + Work Queue com 2 trabalhadores - **1 worker/host**
 - Solução proposta com 11 trabalhadores
 - Solução proposta com 2 trabalhadores
 - Solução proposta com a cache vazia (11 trabalhadores)
 - Solução proposta com a cache vazia (2 trabalhadores)
- Cada cenário foi testado para os workflows BLAST e HECIL

Experimentos e Resultados

Workflow BLAST

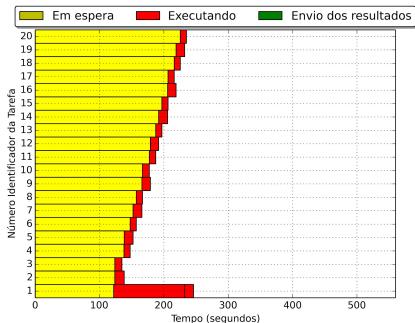
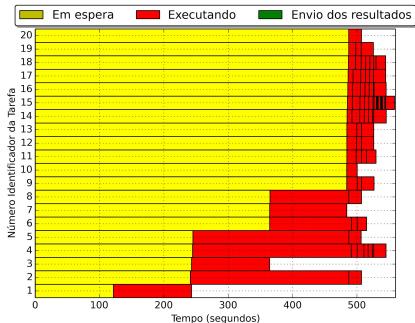


Workflow HECIL

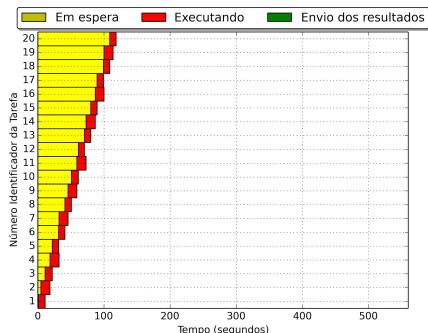
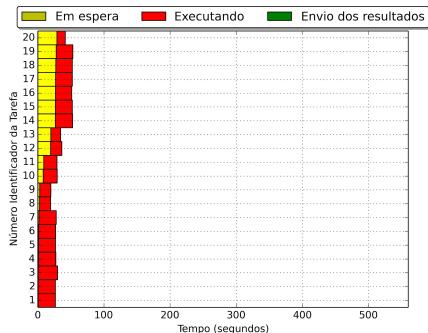


Experimentos e Resultados - BLAST

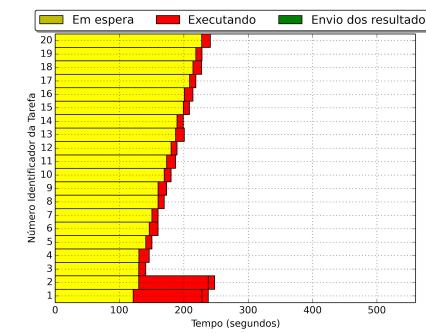
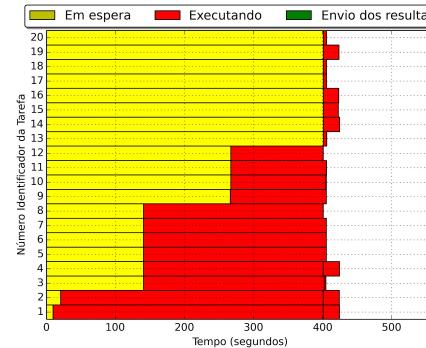
Work Queue original



Work Queue modificado



Cache Vazia

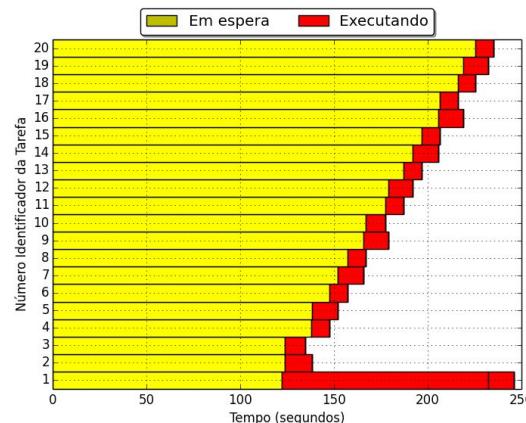
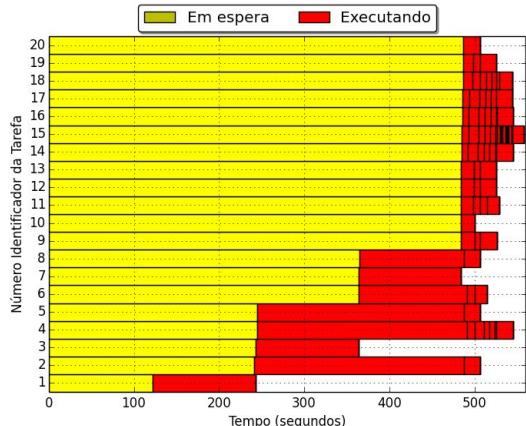


11

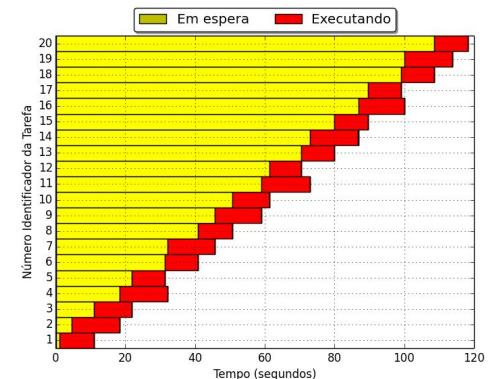
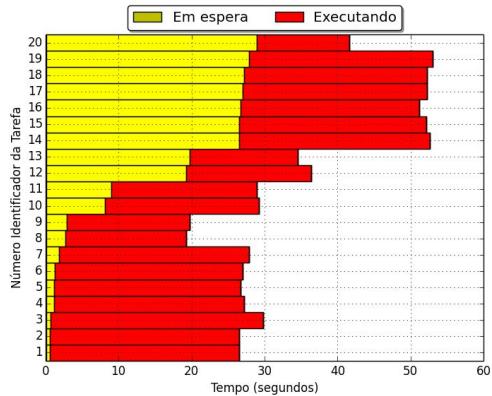
2

Experimentos e Resultados - BLAST

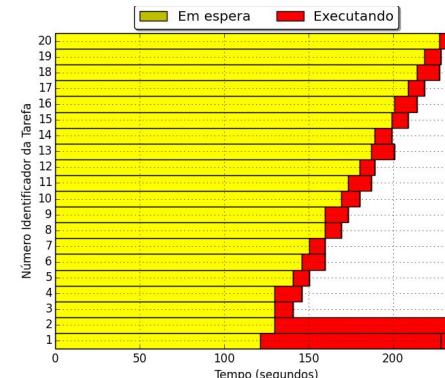
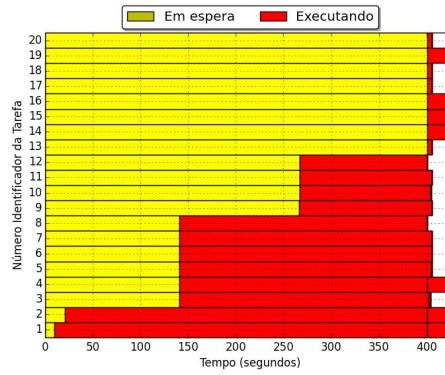
Work Queue original



Work Queue modificado



Cache Vazia

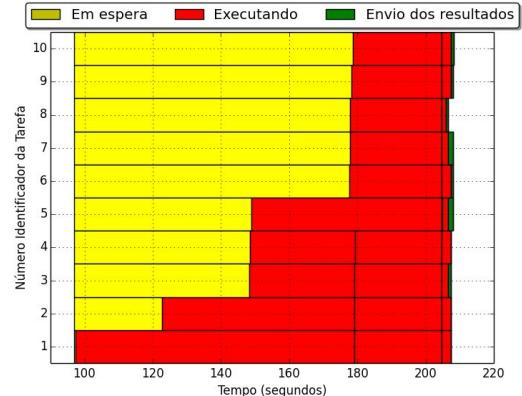


11

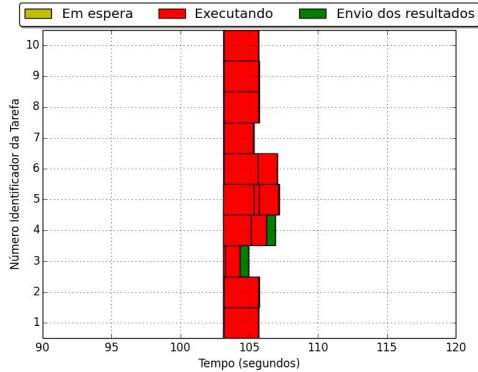
2

Experimentos e Resultados - HECIL

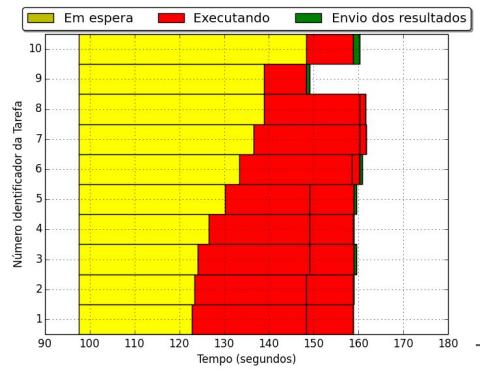
Work Queue original



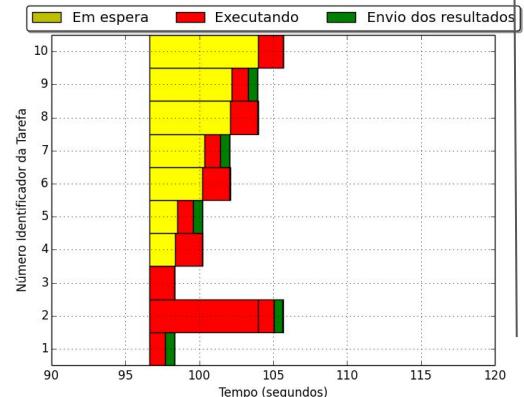
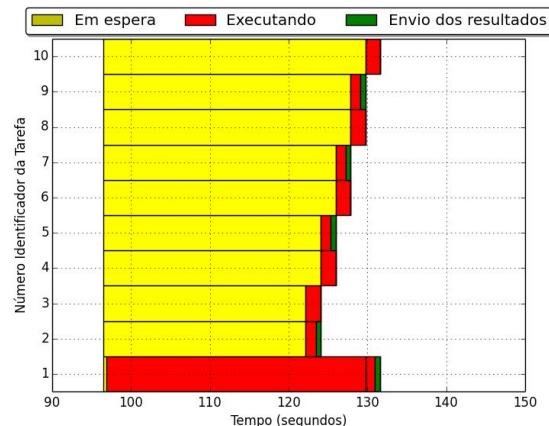
Work Queue modificado



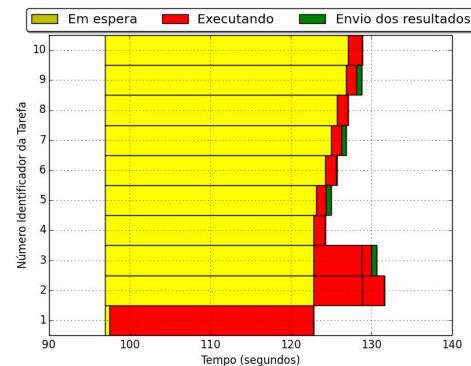
Cache Vazia



11



2



23

Considerações Finais

- Um workflow científico possui diversas tarefas e cada uma delas possui dependências como bibliotecas, executáveis e arquivos de entrada.
- Cientistas enfrentam dificuldades na determinação de um ambiente sólido.
- Integração de contêineres e Workflows diminuem as barreiras para a pesquisa.
- Dentre as vantagens na abordagem proposta:
 - Flexibilidade
 - Reprodutibilidade
 - Escalabilidade

Referências

ALBRECHT, M. et al. Makeflow: a portable abstraction for data intensive computing on clusters, clouds, and grids. In: ACM SIGMOD WORKSHOP ON SCALABLE WORKFLOW EXECUTION ENGINES AND TECHNOLOGIES, 1., New York, NY, USA. Proceedings. ACM, 2012. p.1:1-1:13. (SWEET '12).

STRACE. Linux Syscall Traces. [S.l.: s.n.], 2019 (acesso em Setembro, 2019). <https://strace.io/>.

Skamarock, W. C., J. B. Klemp, J. Dudhia, D. O. Gill, Z. Liu, J. Berner, W. Wang, J. G. Powers, M. G. Duda, D. M. Barker, and X.-Y. Huang, 2019: A Description of the Advanced Research WRF Version 4. NCAR Tech. Note NCAR/TN-556+STR, 145 pp. [doi:10.5065/1dfh-6p97](https://doi.org/10.5065/1dfh-6p97)

Docker. Docker Inc. 2019 (acesso em Setembro, 2019). <https://www.docker.com/>.

StackOverflow. . 2019 (acesso em Setembro, 2019). <https://stackoverflow.com/>.

GitHub. Docker Inc. 2019 (acesso em Setembro, 2019). <https://github.com/>.

BitBucket. 2019 (acesso em Setembro, 2019). <https://bitbucket.org/>.

GitLab. 2019 (acesso em Setembro, 2019). <https://gitlab.com/>.