Darshan

Darshan

Ferramenta escalável para caracterização das operações I/O

Laboratório Nacional de Argonne

Investigação ou tunning de aplicações

Aplicações que utilizam mais de 25% do wallclok com operações de I/O merecem uma atenção especial.

http://www.mcs.anl.gov/research/projects/darshan/

Darshan

Características:

- Baixo overhead
- MPI-IO e POSIX
- Mantém os dados coletados em um buffer de memória
- Não armazena informações completas
- Armazena os dados compactados
- Integração transparente com a maioria das aplicações

Darshan

- darshan-runtime
  Conjunto de bibliotecas e wrappers de compilação para interceptar e contabilizar as chamadas de sistema para acesso aos arquivos.

- darshan-utils
  Conjunto de ferramentas para analisar os dados coletados

darshan-runtime

Coleta informações das aplicações MPI

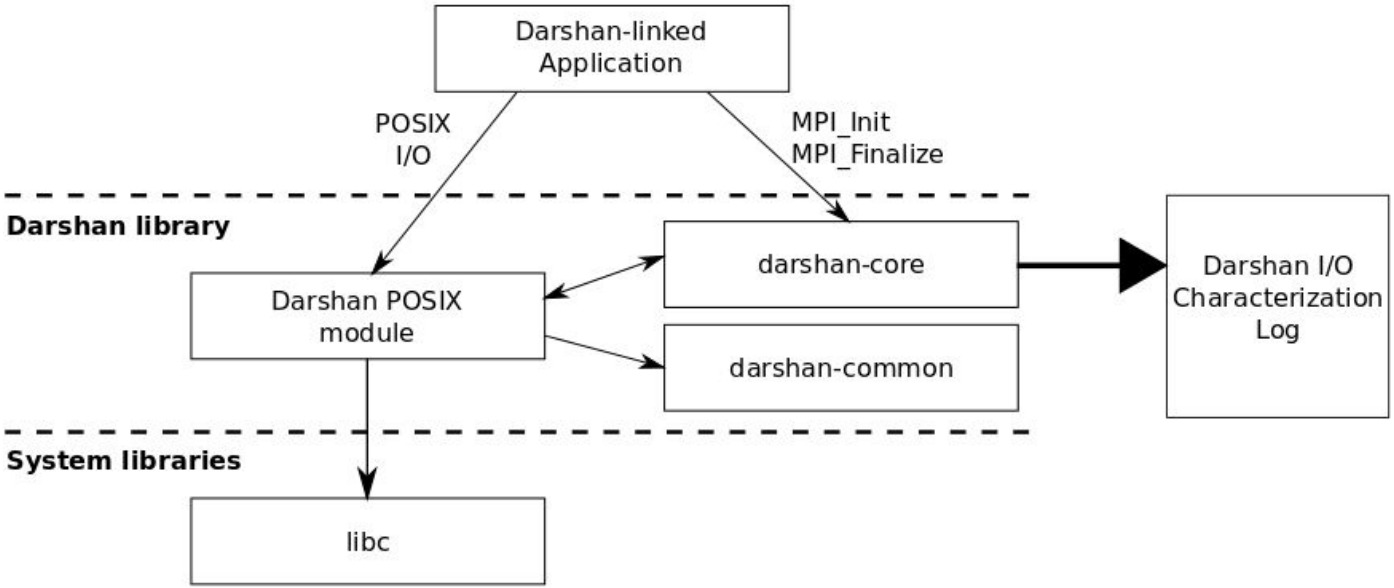
- Inicia a captura com a chamada MPI_Init()
- Encerra com a chamada MPI_Finalize()

Captura informações tanto para acessos POSIX quanto MPI-IO

Informações para HDF5 e PnetCDF são limitadas

Possui uma API que permite implementar novas funcionalidades

darshan-runtime



Darshan

Modo instrumentado

- Necessita recompilar as aplicações
- Utilização de wrappers para os compiladores MPI
- MPI Profile

Modo não instrumentado

- Não é necessário recompilar a aplicação
- Funciona com o pré-carregamento da biblioteca através da variável de ambiente LD_PRELOAD

darshan-utils

Conjunto de ferramentas para visualizar os resultados coletados:

- darshan-job-summary.pl
- darshan-summary-per-file.sh
- darshan-parser
- darshan-analyser
- darshan-convert

darshan-utils

darshan-job-summary.pl

- Gera um resumo, em PDF, sobre as informações do job organizando os dados em tabelas e gráficos.
- O arquivo de saída tem o mesmo nome do arquivo do darshan, mas pode ser escolhido um nome diferente com o parâmetro *--output*

darshan-summary-per-file.sh

- Gera um arquivo pdf para cada arquivo que foi acessado durante a execução do job.

darshan-utils

darshan-parser

- --all : exibe todas as informações
- --base : exibe os dados de log do darshan (default)
- --file : exibe informações referentes a todos os arquivos
- --file-list : informações de cada arquivo acessado
- --perf : informações referentes a performance (unique and shared files)
- --total : totalização de todas as estatísticas coletadas

Utilização:

```
$ darshan-parser [opção] arquivo-de-log.darshan.gz
```

darshan-utils

darshan-analyzer

- Exibe um resumo com o método de acesso usado em todos os arquivos de log de um determinado diretório

```
log dir: /diretorio/de/arquivos/do/darshan
total logs: 14
          shared file access: 0.853211 [4]
    file-per-proccess access: 0.091743 [2]
                mpio access: 0.009174 [8]
             pnetcdf access: 0.000000 [0]
                hdf5 access: 0.000000 [0]

I/O percentage of runtime:
     0.00-10.00: 9
    10.20-20.00: 5
```

darshan-utils

darshan-convert

Utilizado para converter arquivos de formatos antigos no formato atual e também permite remover as informações que identificam os arquivos assim como adicionar informações ao cabeçalho dos arquivos de log.

Limitações

Intel MPI

- Os módulos do Darshan para Intel MPI (instrumentado e não instrumentado) funcionam apenas com executáveis C em C++, não tendo suporte para executáveis Fortran.
- Não gera os arquivos de log quando habilita as variáveis de ambiente para indicar que o filesystem é o Lustre

```
I_MPI_EXTRA_FILESYSTEM=on
I_MPI_EXTRA_FILESYSTEM_LIST=lustre
```

Utilização

Módulos de ambiente

Carregar o módulo bullxde/3.1 (bullx Development Environment)

- darshan/2.3.0_bullxmpi_gnu_[inst|noinst]
- darshan/2.3.0_bullxmpi_intel_[inst|noinst]
- darshan/2.3.0_intelmpi_[inst|noinst]

Utilização

Modo instrumentado

- darshan/2.3.0_bullxmpi_gnu_inst
- darshan/2.3.0_bullxmpi_intel_inst
- darshan/2.3.0_intelmpi_inst

Wrappers para os compiladores MPI:

mpicc.darshan	mpif77.darshan	mpiicpc.darshan
mpiCC.darshan	mpif90.darshan	mpiicc.darshan

Utilização

Modo não instrumentado

- darshan/2.3.0_bullxmpi_gnu_noinst
- darshan/2.3.0_bullxmpi_intel_noinst
- darshan/2.3.0_intelmpi_noinst

Exporta a variável de ambiente LD_PRELOAD automaticamente.

Utilização

Variável de ambiente DARSHAN_LOGPATH

- Antes de executar a aplicação é necessário configurar a variável de ambiente DARSHAN_LOGPATH com o caminho onde os arquivos de log serão armazenados.

Ex:

```
export  DARSHAN_LOGPATH=${SCRATCH}/darshan_logs
```

Utilização

Exemplo do script de submissão

```
...
module load bullxde/3.1
module load bullxmpi_gnu/bullxmpi_gnu-1.2.8.4
module load darshan/2.3.0_bullxmpi_gnu_ [inst|noinst]

# Exportar variável $DARSHAN_LOGPATH
export DARSHAN_LOGPATH=$SLURM_SUBMIT_DIR

...

srun --resv-ports -n $SLURM_NTASKS $EXEC
```

Utilização

Visualizando os resultados

darshan-job-summary.pl /path/to/logfile

Ex:

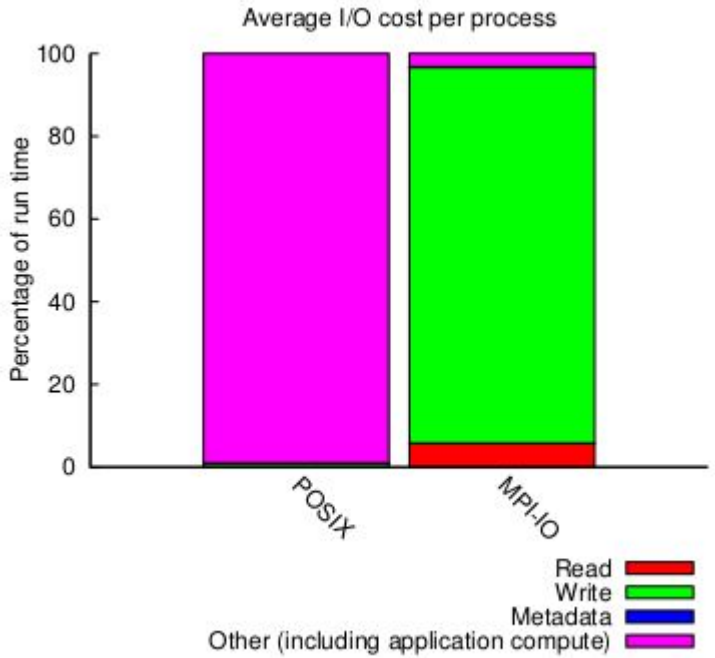
```
Slowest unique file time: 1240.9816

Slowest shared file time: 0

Total bytes read and written by app (may be incorrect): 7215545057280

Total absolute I/O time: 1240.9816
```

Analisando os resultados

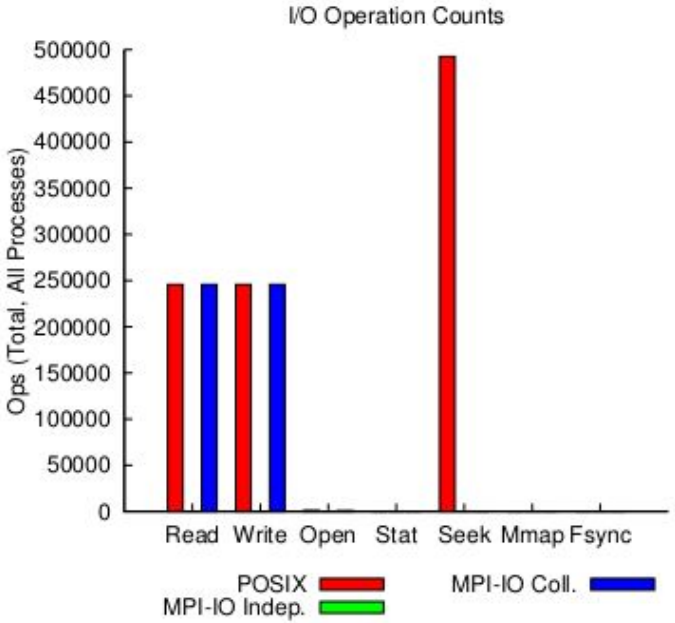


Average I/O cost per process

Quanto tempo da aplicação foi utilizado em operações de I/O em cada API

Em sistemas com poucos servidores de metadados uma quantidade muito elevada dessas operações podem ser serializadas e reduzir a performance.

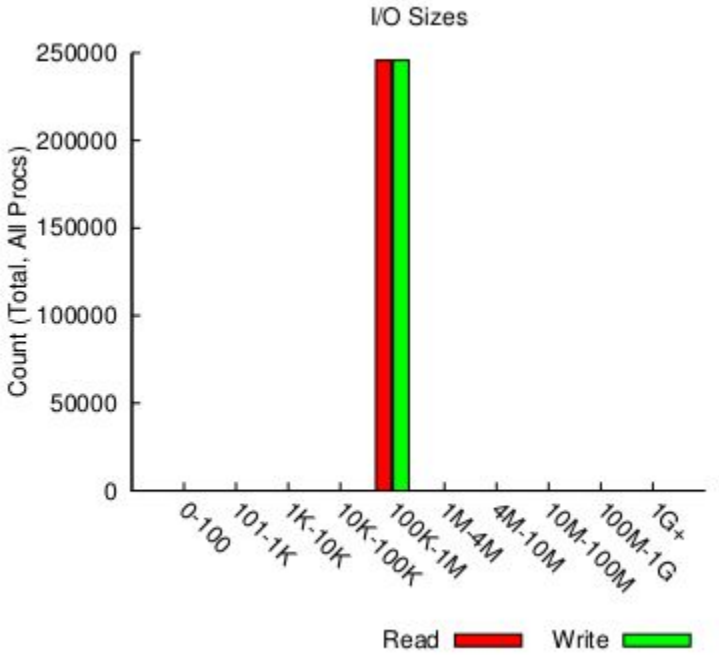
Analisando os resultados



I/O Operation Counts

Exibe quantas operações de I/O de cada tipo ocorreram.

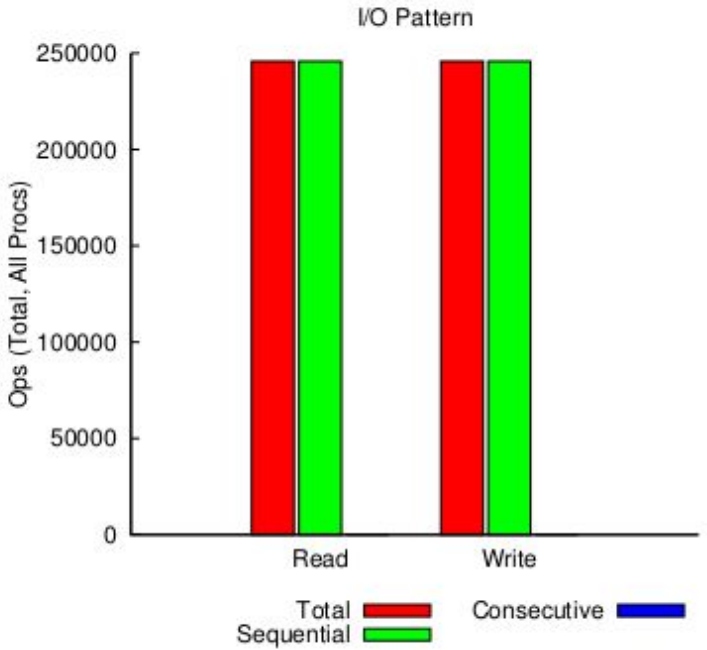
Analisando os resultados



I/O Sizes

Exibe o quantidade operações realizadas e seus respectivos tamanhos para leitura e escrita.

Analisando os resultados



I/O Pattern

Informações sobre o padrão de acesso aos arquivos.

Consecutivas: o acesso aos blocos de dados é feito sem gaps.

Sequencial: o acesso aos blocos de dados é feito de forma irregular.

Lab 1

Compilando IOR no modo instrumentado

```
$ module load bullxde/3.1
$ module load bullxmpi_gnu/bullxmpi_gnu-1.2.8.4
$ module load darshan/2.3.0_bullxmpi_gnu_ inst

$ tar zxvf IOR-2.10.3.tgz
$ mv IOR IOR-inst
$ cd IOR-inst/src/C
```

Editar o arquivo Makefile.config e alterar a diretiva **CC.Linux** para:

CC.Linux = mpicc.darshan

Compilar:

```
$ gmake mpiio
```

Lab 1

Altere as variáveis do script *ior-inst.srm* para que reflitam as configurações do seu ambiente:

```
BASEDIR="${SCRATCH}/mc03"
CONFIGFILE=${BASEDIR}/IOR.config
OUTDIR=${BASEDIR}/output
EXEC=${BASEDIR}/IOR-inst/src/C/IOR
export DARSHAN_LOGPATH="${BASEDIR}/darshan"
```

Submeta o script com o comando sbatch:

```
$ sbatch ior-inst.srm
```

Lab 2

Compilando IOR no modo não instrumentado

```
$ module load bullxde/3.1
$ module load bullxmpi_gnu/bullxmpi_gnu-1.2.8.4
$ module load darshan/2.3.0_bullxmpi_gnu_ noinst

Verificar o conteúdo da variável LD_PRELOAD
$ echo $LD_PRELOAD

$ tar zxvf IOR-2.10.3.tgz
$ mv IOR IOR-noinst
$ cd IOR-noinst/src/C

Compilar:
$ gmake mpiio
```

Lab 2

Altere as variáveis do script *ior-noinst.srm* para que reflitam as configurações do seu ambiente:

```
BASEDIR="${SCRATCH}/mc03"
CONFIGFILE=${BASEDIR}/IOR.config
OUTDIR=${BASEDIR}/output
EXEC=${BASEDIR}/IOR-noinst/src/C/IOR
export DARSHAN_LOGPATH="${BASEDIR}/darshan"
```

Submeta o script com o comando sbatch:

```
$ sbatch ior-noinst.srm
```

Lab 3

Analisando os resultados:

- Gere um arquivo PDF para cada resultado do darshan

```
$ darshan-job-sumary.pl logfile
```

- Utilizando o comando darshan-parser obtenha o tempo total gasto com as operações de leitura e escrita

```
$ darshan-parser --total logfile | grep BYTES_READ
$ darshan-parser --total logfile | grep BYTES_WRITTEN
```

Lab 4

Altere os scripts dos Labs anteriores e realize execuções do IOR variando o stripe count e stripe_size do Lustre, modifique também os parâmetros transferSize, segmentCount e blockSize para simular outros comportamentos para a aplicação:

```
lfs getstripe filename|dirname

lfs setstripe -s stripe_size -c stripe_count filename|dirname
```