Shrinkwrap: Creating HPC Containers

Tim Shaffer, Nick Hazekamp

CCTools

UNIVERSITY OF NOTRE DAME

What is Shrinkwrap?

New tool for exporting CVMFS repositories to a POSIX FS

Released as part of CernVM-FS 2.6.0

Officially supported tool

Built on Libcvmfs implementation, configuration

CCTools

UNIVERSITY OF NOTRE DAME

Use Case: HPC Environments

Experiments are already working on taking advantage of HPC resources (e.g. at NERSC, CSCS)

These sites often can't use the default setup:

- No FUSE
- Limited network access
- No administrative or infrastructure changes

Admins recommend containers for distributing software

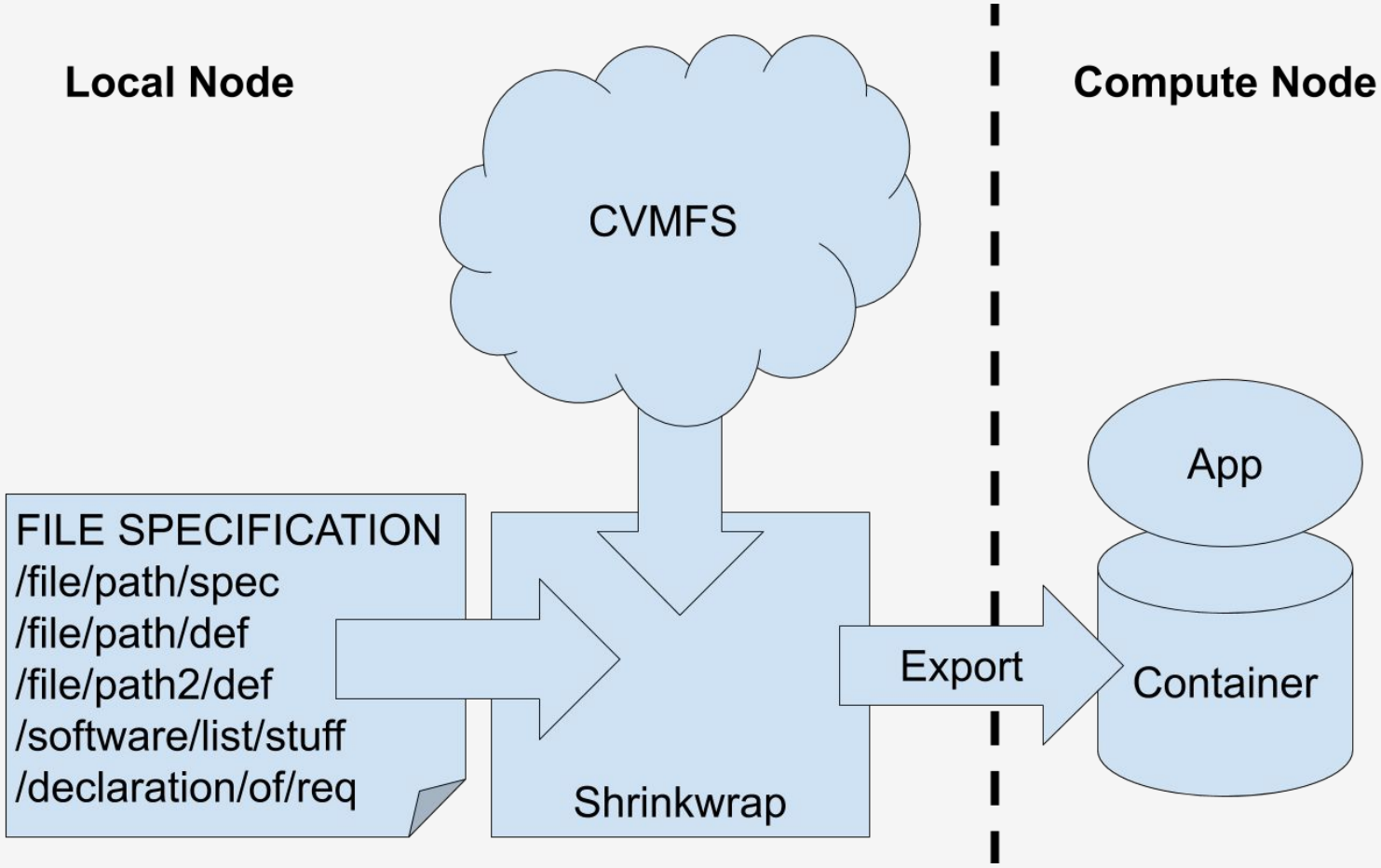
Use Case: HPC Environments

Shrinkwrap can efficiently export (subsets of) CVMFS repos, which can be used for

- container images
- shared FS copies
- tarballs, SquashFS images, etc.

Designed as a lightweight tool to fit into different workflows, with **unprivileged operation** and **efficient updates**

Use Case: HPC Environments



CCTools

UNIVERSITY OF NOTRE DAME

Use Case: Benchmarking

Network jitter, cache effects, etc. can make performance measurements difficult for CVMFS applications

We can use Shrinkwrap + a container technology to build **self-contained benchmark jobs** with **low setup cost**

More info on current benchmarking work

CCTools

UNIVERSITY OF NOTRE DAME

Current Approaches

UnCVMFS is an external tool for downloading an entire repo. Initial download is expensive, but unCVMFS reads catalogs to quickly find changes on update.

rsync can be used to copy out of a mounted repo. Uses existing config and downloads more selectively, but runs into issues with detecting changes, deduplication.

Getting Shrinkwrap

Packages are currently available for Red Hat and Debian based platforms!

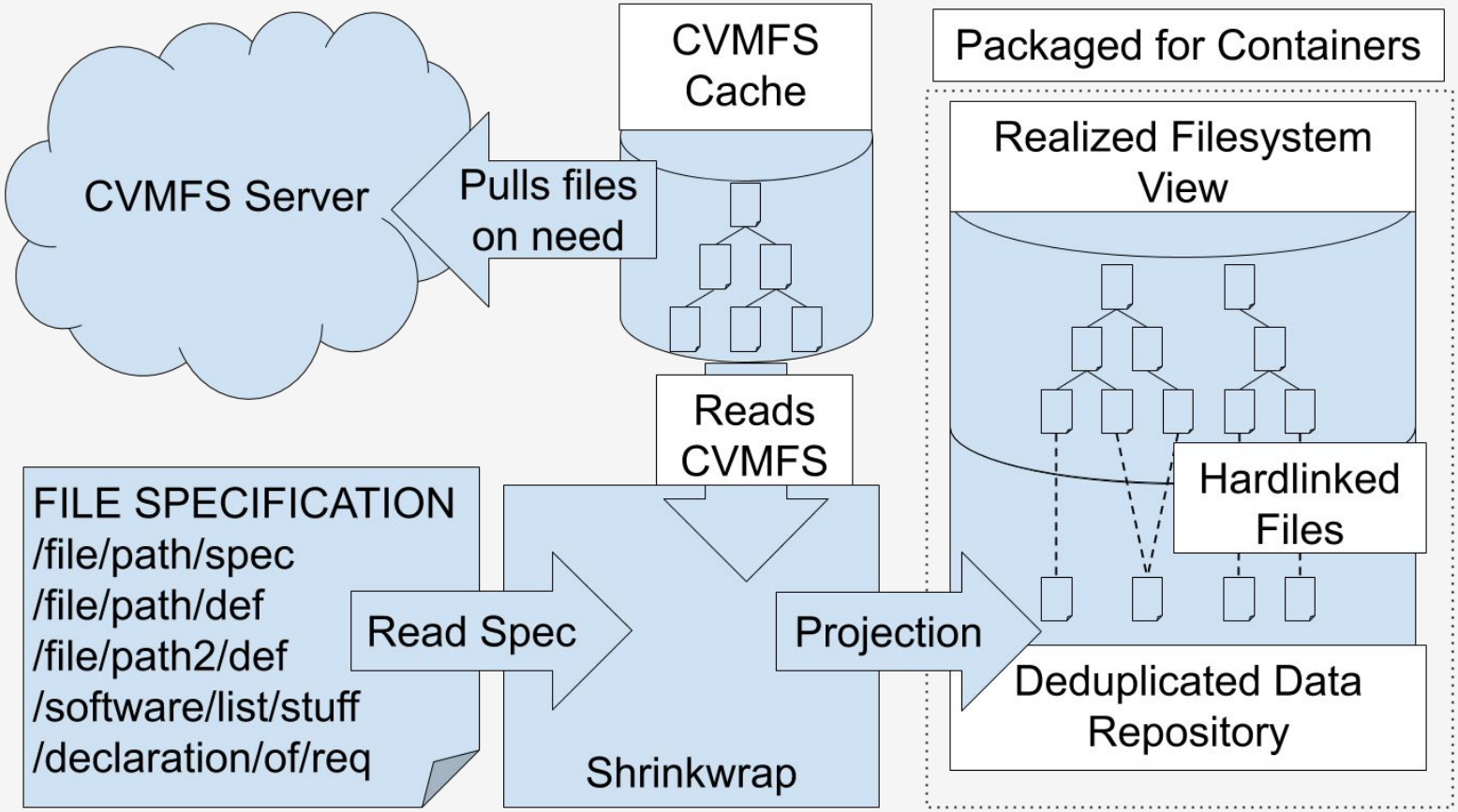
Install the `cvmfs-shrinkwrap` package using your favorite package manager.

If building from source, pass `-DBUILD_SHRINKWRAP=on` to cmake.

CCTools

UNIVERSITY OF NOTRE DAME

Shrinkwrap Design



CCTools

UNIVERSITY OF NOTRE DAME

Shrinkwrap Design

Checked out repos contain:

- File trees under `<fqrn>/<path>`
- Content-addressed files under `.data`
- Provenance info under `.provenance`

Must be on the same filesystem (hard links)

NOTE: CVMFS variables in symlinks are resolved at image creation time, not at run time

CCTools

UNIVERSITY OF NOTRE DAME

Configuration

```
CVMFS_REPOSITORIES=sft.cern.ch
CVMFS_REPOSITORY_NAME=sft.cern.ch
CVMFS_CONFIG_REPOSITORY=cvmfs-config.cern.ch
CVMFS_SERVER_URL='http://cvmfs-stratum-zero-hpc.cern.ch/cvmfs/sft.cern.ch'
CVMFS_CACHE_BASE=/var/lib/cvmfs/shrinkwrap
CVMFS_HTTP_PROXY=DIRECT                          # 1
CVMFS_KEYS_DIR=/etc/cvmfs/keys/cern.ch           # 2
CVMFS_SHARED_CACHE=no                            # 3
CVMFS_USER=cvmfs
```

1. Adjust to your site
2. Need to be provided for shrinkwrap
3. Important as libcvmfs does not support shared caches

CCTools

UNIVERSITY OF NOTRE DAME

Unprivileged Operation

Add UID/GID mapping files to your config file:

```
CVMFS_UID_MAP=uid.map
CVMFS_UID_MAP=gid.map
```

Then fill in `uid.map` and `gid.map` with your UID/GID:

```
* 1000
```

Running Shrinkwrap

We're (almost) ready to run!

```
$ cvmfs_shrinkwrap \
    --repo sft.cern.ch \
    --src-config sft.config \
    --spec-file job.spec \
    --dest-base $OUTDIR \
    --threads 16
```

CCTools

UNIVERSITY OF NOTRE DAME

Writing Specifications

Full Repo Specification

```
/*
```

Specification Format

Specific file:

```
/lcg/releases/gcc/7.1.0/x86_64-centos7/setup.sh
```

Full directory tree:

```
/lcg/releases/ROOT/6.10.04-4c60e/x86_64-cenots7-gcc7-opt/*
```

Exclusion:

```
!/lcg/releases/uuid
```

CCTools

UNIVERSITY OF NOTRE DAME

Example Specification

ROOT version 6.10 (~8.3 GB)

```
/lcg/releases/ROOT/6.10.04-4c60e/x86_64-centos7-gcc7-opt/*
/lcg/contrib/binutils/2.28/x86_64-centos7/lib/*
/lcg/contrib/gcc/*
/lcg/releases/gcc/*
/lcg/releases/lcgenv/*
```

Static Specifications

Specifications can be difficult to write by hand:

- Full dependencies might not be obvious.
- Must keep in sync with updates.
- Easy to forget something (breakage) or include more than necessary (bloat).

Static specifications can be necessary in some situations, e.g. non-deterministic or data-dependent component loading.

Dynamic Tracing

Alternatively, get specifications by tracing individual jobs:

- Can be done automatically
- No need for application/version specific knowledge
- Larger specifications (usually >*O*(1000) of lines)

Traces only capture individual past jobs, might not work for future jobs!

Tracing via FUSE Client

Enable tracing for a mounted repo by adding a config option:

```
CVMFS_TRACEFILE=/tmp/cvmfs-trace-@fqrn@.log
```

The trace log is buffered, so be sure flush before and after running the job:

```
$ cvmfs_talk tracebuffer flush
```

CCTools

UNIVERSITY OF NOTRE DAME

Tracing with Parrot

Parrot can record file accesses by a command when run with an extra option:

```
--namelist <LOG>
```

No config changes or administrative privileges required.

Building Containers

SquashFS Image

Repo copy can be exported to a variety of formats: tarball, zip, disk image, etc.

SquashFS is a good choice:

- Data and metadata compression
- Userland creation tools
- Directly mountable

```
$ mksquashfs /PATH/TO/REPO image.squashfs
```

CCTools

UNIVERSITY OF NOTRE DAME

Embedding vs. Bind Mounting

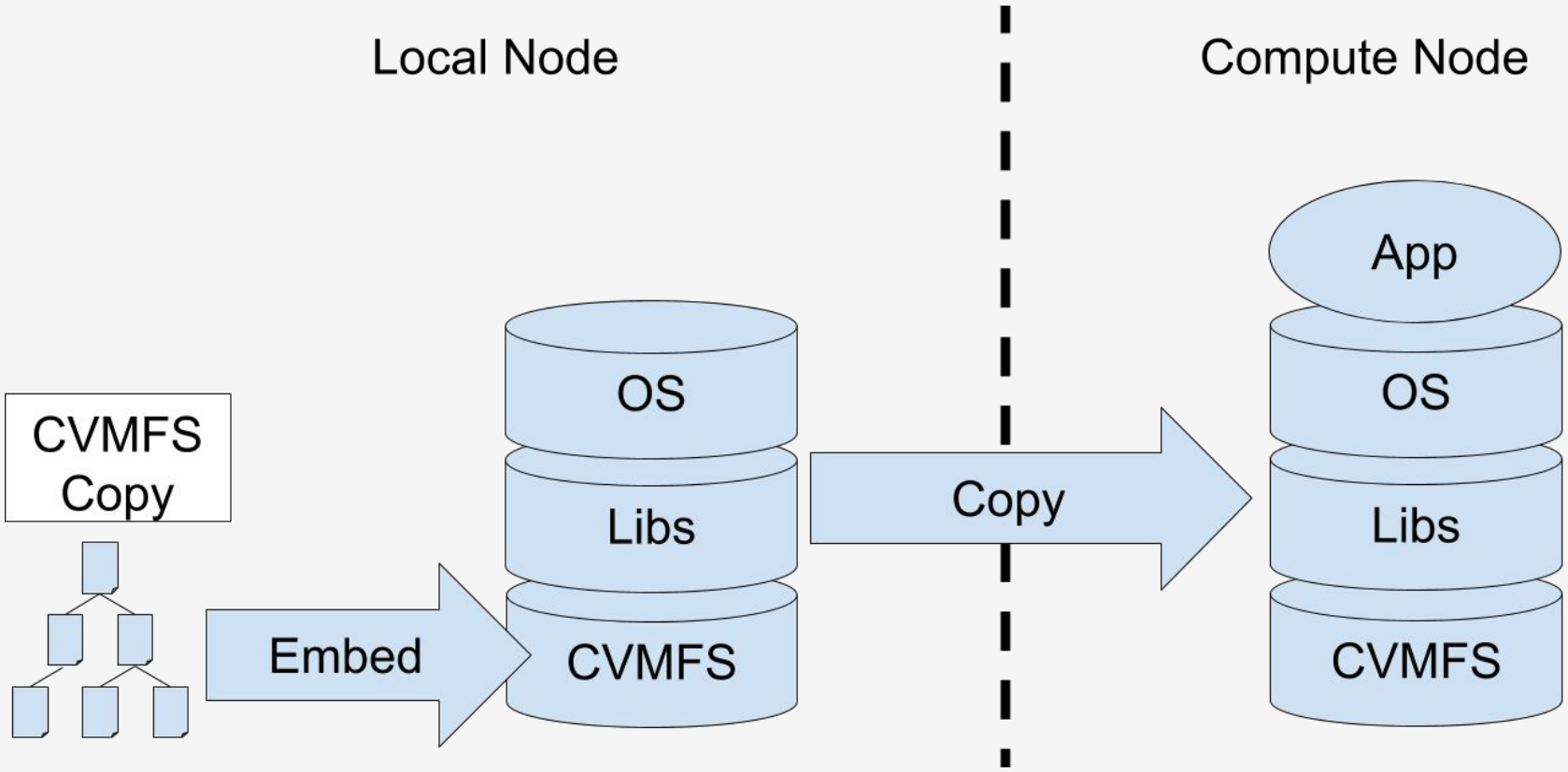
For container images (Docker, Shifter, Singularity, etc.) we can **embed** CVMFS data directly in the image

- Self-contained images.
- Build time and image size can be a problem.

or **bind mount** the repo from outside at run time.

- Use shared copy on local disk, parallel FS, etc.
- Image only works if repo is available at compute nodes.

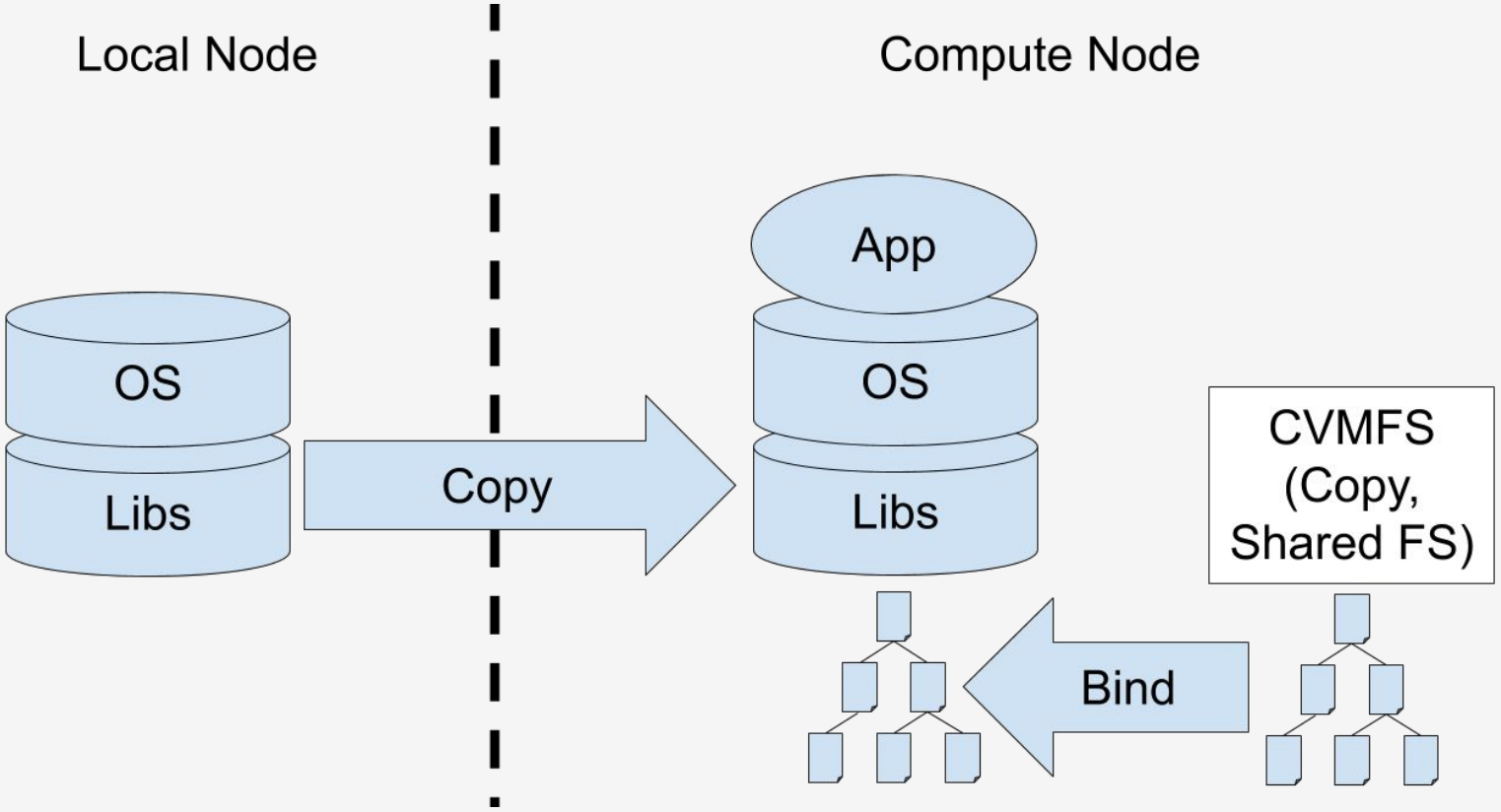
Embedding Repo in Container



CCTools

UNIVERSITY OF NOTRE DAME

Bind Mounting into Container



CCTools

UNIVERSITY OF NOTRE DAME

Docker Example

Copy an existing repo on the host

```
COPY --chown=cvmfs:cvmfs /PATH/TO/REPO /cvmfs
```

or run Shrinkwrap *inside* the container during build

```
RUN cvmfs_shrinkwrap --dest-base /cvmfs  [...more options]
```

Make sure a previous step installed Shrinkwrap!

Same, for Singularity

Copy from host:

```
%files
    /PATH/TO/REPO /cvmfs
```

Download during container build:

```
%post
    cvmfs_shrinkwrap --dest-base /cvmfs [...more options]
```

CCTools

UNIVERSITY OF NOTRE DAME

Bind Mounting

Docker and Singularity can both bind in directories **at run time** from the host.

Docker: `--volume /PATH/TO/REPO:/cvmfs`

Singularity: `--bind /PATH/TO/REPO:/cvmfs`

Execution nodes must be able to access repo copies!

Using Shifter

NERSC uses Shifter for managing containers.

Import (and flatten) a Docker image:

```
$ shifterimg -v pull docker:image_name:latest
```

To bind mount, add to your submit script:

```
#SBATCH --volume="/PATH/TO/REPO:/cvmfs"
```

Thank you

Contact info

tshaffe1@nd.edu, nhazekam@nd.edu

Documentation links

https://cvmfs.readthedocs.io/en/stable/cpt-shrinkwrap.html

https://cvmfs.readthedocs.io/en/stable/cpt-tracer.html

See what else we're doing

http://ccl.cse.nd.edu/

CCTools

UNIVERSITY OF NOTRE DAME