gem5 at home (or work/school)

Getting help

gem5 has lots of resources to get help:

1. Documentation at gem5 doxygen
2. Ways to reach out for help:
    - Github discussions **This is the main place for questions**
    - gem5 Slack channel
    - Join our mailing lists:
        - gem5-dev@gem5.org : For discussions regarding gem5 development
        - gem5-users@gem5.org : For general discussions about gem5 and its use
        - gem5-announce@gem5.org : For general gem5 announcements
3. Youtube videos

These links and more information are also available at https://www.gem5.org/ask-a-question/

> We do our best to get to questions, but they often go unanswered. This isn't because it's not a good question, but because we don't have enough volunteers.

Running gem5 at home

- gem5 performance qualities
  - Single threaded
  - Consumes lots of RAM (if you want to model 32 GB of memory, it needs 32 GB of memory to model it)
  - Can take a lot of time
- Because of this its best to run multiple experiments in parallel
- Recommended hardware:
  - High single thread performance
  - Doesn't need many cores
  - LOTS OF RAM

System software requirements

- Ubuntu 22.04+ (at least GCC 10)
  - 20.04 works, but there are bugs in GCC 8 (or 9, whatever the default is) and you have to upgrade the GCC version.
- Python 3.6+
- SCons
- Many optional requirements.

This *should* work on most Linux systems and on MacOS.

See our Dockerfiles for the most up-to-date version information:

`gem5/util/dockerfiles/`

Using dockerfiles

If you have trouble, we have docker images.

Here's a generic docker command that should work.

```
docker run --rm -v $(pwd):$(pwd) -w $(pwd) ghcr.io/gem5/ubuntu-24.04_all-dependencies:v24-0 <your command>
```

- Runs the image at `https://ghcr.io/gem5/ubuntu-24.04_all-dependencies:v24-0`.
- Automatically removes the docker image (`--rm`)
- Sets it up so that the current directory (`-v $(pwd):$(pwd)`) is available inside the docker container
- Sets the working directory to the current directory (`-w $(pwd)`)
- Runs a command.
- Every command will now need to run with this to make sure the libraries are set up correctly.

> I cannot **strongly enough** emphasize that you should not run interactively in the docker container. Use it to just run one command at a time.

The devcontainer

The devcontainer we've been using is based off of `ghcr.io/gem5/ubuntu-24.04_all-dependencies:v24-0`, but also includes some gem5 binaries.

You can find it at `ghcr.io/gem5/devcontainer:bootcamp-2024`.

The source is at `gem5/utils/dockerfiles/devcontainer`.

Recommended practices

- Unless planning on contributing to gem5 or you need to use recently developed work, use the `stable` branch.
- Create branches off of stable.
- Don't modify parameters of python files in `src/`. Instead create *extensions* of stdlib types or SimObjects.
- Don't be afraid to read the code. The code is the best documentation.

gem5 Cheat Sheet

Downloading

```
git clone https://github.com/gem5/gem5
```

Building

```
scons build/ALL/gem5.opt -j$(nproc)
```

Customizing the build

```
scons menuconfig build/ALL
```

Running

```
build/ALL/gem5.opt [gem5 options] <your script> [your script options]
```

Example scripts

See `configs/examples/gem5_library`

Resources/Workloads/Disk Images/Suites

https://resources.gem5.org/

`obtain_resource(<resource id>)`

Debugging

```
build/ALL/gem5.opt --debug-flags=<debug flags> <your script>
build/ALL/gem5.opt --debug-help
```

Stats

Found in `m5out/stats.txt`

```
simulator.get_simstats()
```

Full system

```
util/term/m5term localhost 3456
```

More cheat sheet

Creating a disk image

See https://github.com/gem5/gem5-resources/. Use the packer files already available or create your own.

```
packer build <packer json file>
```

Take a checkpoint

```
sim.save_checkpoint('checkpoint')
```

Restore a checkpoint

```
sim = Simulator(checkpoint_path='checkpoint')
```

Controlling simulation

```
def my_exit_handler():
    yield False #continue simulation
    yield True #stop simulation
sim  = Simulator(on_exit_event={
  ExitEvent.WORKBEGIN: my_exit_handler})
```

Stdlib components

- `Board`: Connects things together
- `Processor`: Multiple cores
- `CacheHierarchy`: The caches. Either classic or Ruby
- `MemorySystem`: The main memory

Final things

Big thanks

Bobby Bruce

Ivana Mitrovic

Harshil Patel

Erin Le

Mahyar Samani

Zhantong Qiu

William Shaddix

Samuel Thomas

Yuyi Li

Leo Redivo

Mitha Mysore

Alyssa Vallejo

Noah Krim

Saili Karkare

Matt Sinclair

Big thanks to you all!

Please let us know how we did:

https://forms.gle/M6HZHxGjXpcdw4kZ8

Reach out to us:
Jason Lowe-Power jlowepower@ucdavis.edu
Tamara Silbergleit Lehman
tamara.lehman@colorado.edu