# SKaMPI 5 for the impatient

Thomas Worsch

April 16, 2008

## 1 Source file

The sources for SKaMPI 5 are provided in a single `gzip`ped `tar` file. Release numbers for SKaMPI 5 are of the form `5.`$\langle x\rangle$`.`$\langle y\rangle$ where $\langle x\rangle$ is the minor version number of SKaMPI 5 (e.g. `0`) and $\langle y\rangle$ is the sub-release number (currently both are always single digts).

The initial release of `skampi-5.`$\langle x\rangle$`.`$\langle y\rangle$ is always provided in a file named `skampi-5.`$\langle x\rangle$`.`$\langle y\rangle$`.tar.gz` .

If at a later time an updated version with some small bug fixes applied but without any new features is released, the file name will be of the form `skampi-5.`$\langle x\rangle$`.`$\langle y\rangle$`-r`$\langle rev\rangle$`.tar.gz` where $\langle rev\rangle$ is a four digit revision number (corresponding to our svn revision number).

## 2 Unpacking

```
gzip -cd skampi-5.⟨x⟩.⟨y⟩.tar.gz | tar xf -
```

or

```
gzip -cd skampi-5.⟨x⟩.⟨y⟩-r⟨rev⟩.tar.gz | tar xf -
```

### Explanations

- Alternatively you may be able to unpack the archive using

  `tar zxf` $\langle file\rangle$`.tar.gz`

- Depending on what kind of tar file you unpack you should get a directory named

  `skampi-5.`$\langle x\rangle$`.`$\langle y\rangle$

  or

  `skampi-5.`$\langle x\rangle$`.`$\langle y\rangle$`-r`$\langle rev\rangle$

  containing the C source file, example input files and the documentation for SKaMPI 5.

For the rest of this short guide we assume a directory name without revision number.

# 3 Compiling SKaMPI 5

```
cd skampi-5.⟨x⟩.⟨y⟩
make
```

### Explanations

- Until now we have only used GNU make. It works for us. If it doesn't work, in particular because you have a different make, please let us know about your problem.
- `make` needs to know how to compile an MPI application. It uses the content of the variable `MPICC` as the compiler for all `.c` files. The default value is `mpicc`. If you have to use a different program for compiling MPI sources, please change it, e.g. by using

  ```
  make MPICC='mympicc'
  ```

- Analogously, if the default `-O2` for the compile flags is not good in your case, try something like

  ```
  make CFLAGS='-O0'
  ```

# 4 Run SKaMPI 5

```
mpiexec -n 4 skampi -i foo.ski -o bar.sko
```

### Explanations

- Depending on your MPI implementation you may need to use another command, e.g. `mpirun` to start an MPI program.
- Depending on your MPI implementation you may need to use another way to specify the number of processes for `MPI_COMM_WORLD`, e.g. `-procs 4`.

# 5 Investigate the output file of SKaMPI 5

```
less .......
gnuplot .......
```

## Explanations

- The format of SKaMPI's output file is intended to be readable by humans (maybe with the exception that you had a run on many many processors). Just try it. Each line represents a measurement with a fixed set of parameters. If you had only one loop around the `measure` statement in the input file, such a line looks like this:

  ⟨*loop.var*⟩`=` ⟨*val*⟩ ⟨*msg.size*⟩ ⟨*run.time in* $\mu s$⟩ ... more fields ...

- The format of SKaMPI's output file is intended to be used easily with `gnuplot`. In the case of one loop as right above you can access the running time by using the construct `using ($4)` in your `plot` statement, because it is the fourth (white-space separated) field in each line.

- There is a Python script called `skompare.py`. It's documentation should be provided in the full manual.