Parallel PARI

B. Allombert

IMB
CNRS/Université Bordeaux 1

26/01/2012

Introduction

We add support for two common mutli-threading technologies :

- POSIX thread : run on a single machine, lightweight, fragile.
- Message passing interface (MPI) : run on as many machine as you want, robust, heavyweight.

An example problem

We want to compute the value of a function for all integers less than 1000. Each call take 1 hour.

```
for(i=1,1000,print(i,":",fun(i)))
```

This will take 1000 hours.

Lignes directrices

The simplest parallel solution

Now assume we have a MPI cluster with 100 cores at our disposal. We rewrite the program as follow :

```
N=eval(getenv("OMPI_COMM_WORLD_RANK"));
for(i=10*N+1,10*N+10,
    write(Str("fun",N,),i,":",fun(i)))
```

We launch it using OpenMPI mpirun command :

```
mpirun -np 100 gp fun.gp
```

Your computation will be finished in 10 hours, the results split in the files fun0 to fun99.

The experimental GIT branch bill-pareval

- New Configure flag : –mt=single, –mt=pthread, or –mt=mpi
- New GP function `pareval`
- Parallel algorithms : SEA, quadclassunit.

Lignes directrices

Using pareval

```
res=pareval(vector(1000,i,()->fun(i)))
for(i=1,1000,print(i,":",res[i]))
```

Lignes directrices

Parallel SEA/quadclassunit

```
\g1
ellap(ellinit([1,3]),nextprime(2^400))
quadclassunit(1-2^140)
```

The libpari interface

- `handle = mt_queue_start(worker)` Return a handle for parallel evaluation of `worker`.
- `mt_queue_submit(handle, work)` Submit `work` to be evaluated by `worker`.
- `result = mt_queue_get(handle, pending)` Return the evaluation by `worker` of some of the previously submitted works. Set pending to the number of remaining pending works.
- `mt_queue_end(void *handle)` Free the ressource allocated by `handle` and end the parallel execution.

Call to `mt_queue_submit` and `mt_queue_get` must be alternated.

Lignes directrices

Example : Code of pareval

```
GEN pareval_worker(GEN i, GEN C)
{
  retmkvec2(icopy(i), closure_callgenall(C, 0));
}
GEN pareval(GEN C)
{
  long l = lg(C), i, pending = 0;
  GEN worker = snm_closure(is_entry("_pareval_worker"));
  void *mt = mt_queue_start(worker);
  GEN V = cgetg(l, t_VEC), done;
  for (i=1; i<l || pending; i++)
  {
    mt_queue_submit(mt, i<l? mkvec2(utoi(i),gel(C,i)));
    done = mt_queue_get(mt, &pending);
    if (done) gel(V,itou(gel(done,1))) = gel(done,2);
  }
  mt_queue_end(mt); return V;
```

Low-level PARI POSIX thread interface

You need to use `Configure -enable-tls`. See Appendix D of the manual, and the file `example/thread.c`
Parent thread :

- `pari_thread_alloc()` Allocate a PARI stack for a thread.
- `pari_thread_free()`

Child thread :

- `pari_thread_start()` Initialize threads using the specified stack.
- `pari_thread_close()`