Two kinds of parallel problems

A simulation run takes about a second.

Want to do 1000's of them,
varying a dozen or so parameters.

A simulation of a large network takes hours.

Want to spread the problem over several machines,
each machine handling a subset of the neurons in the network

Serial

```
s = 0
for i = 1, 10 {
    s += f(i)
}
```

Parallel

```
s = 0
for i = 1, 10 {
    pc.submit("f", i)
}
while (pc.working) {
    s += pc.retval
}
```

Goals

Keep all the machines as busy as possible.

If there is only one machine the parallel program should run as fast as the serial program.

Things asked for earlier tend to get done earlier.

Assumptions

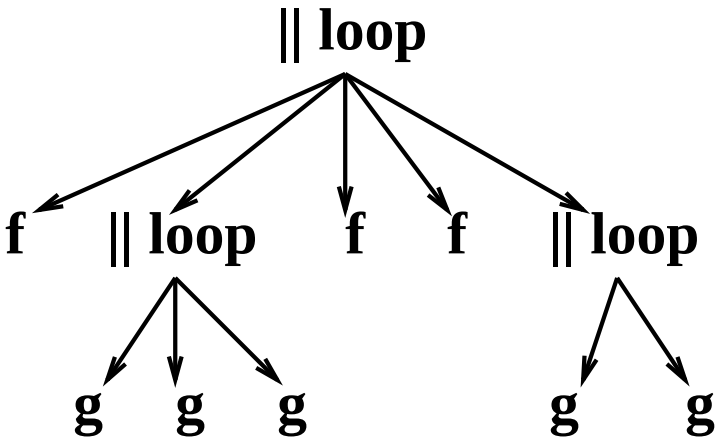
Workstation cluster − 1, 3, 15, 100 machines.

Wide variety of machine speeds.

Sending a byte is much slower than executing a hoc statement.

Domain

Very coarse grain parallelization.



NEURON's style

A bulletin board
. . . on top of MPI.

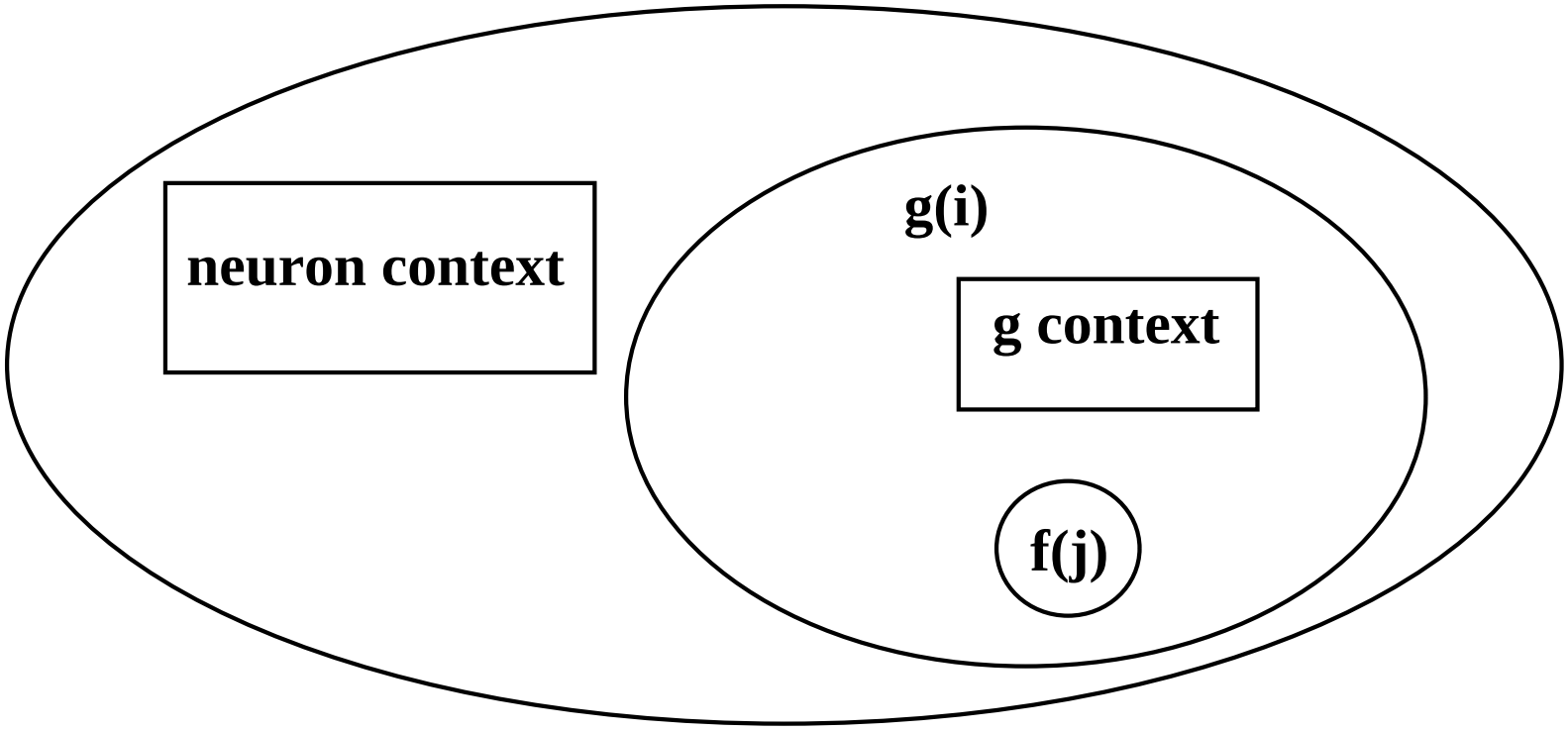
Launching a parallel program

```
objref pc
pc = new ParallelContext()

  // setup which is exactly
  // the same on every machine
  // i.e. declaration of all
  // functions, procedures,
  // setup of neurons

pc.runworker

  // the master scatters tasks
  // onto the bulletin board
  // and gathers results

pc.done
```

Example.hoc

```
objref pc
pc = new ParallelContext()

func f() {local s, i
  s = 0
  for i=1,100000 {
    s += $1
  }
  return s
}

{pc.runworker()}

runtime = startsw()
s = 0
for i=1,10 {
  pc.submit("f", i)
}
while (pc.working()) {
  s += pc.retval
}
print "sum = ", s
print "runtime ", startsw() - runtime
{pc.done()}
quit()
```

```
$ mpiexec -n 1 nrniv -mpi example.hoc
numprocs=1
NEURON -- VERSION 7.2 (428:986821b56b98) 2010-03-17
...
sum = 5500000
runtime 0.079999924

$ mpiexec -n 4 nrniv -mpi example.hoc
numprocs=4
NEURON -- VERSION 7.2 (428:986821b56b98) 2010-03-17
...
sum = 5500000
runtime 0.019999981

$
```



post ⟶ Bulletin board

take
look ⟵ Bulletin board
look_take

context("stmt") : stmt executed on every worker

With Python

```
def f(arg1, arg2):
    ...
    return any_pickleable_object

...
pc.submit(f, (arg1, arg2))
...
while pc.working():
    r = pc.pyret()
```