FASTFLOW

mc-fastflow.sourceforge.net

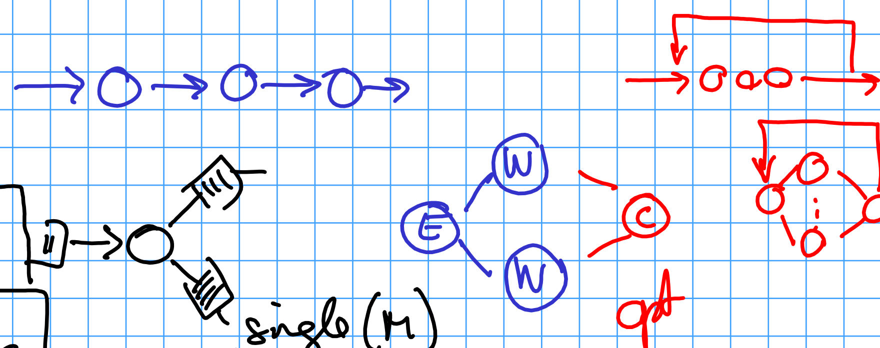
prog. par. strutturata (STREAM PARALLEL)

multicore (shared memory)

FINE GRAIN

PIPELINE FARM FEEDBACK

full C++

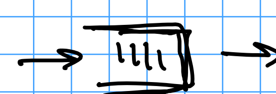


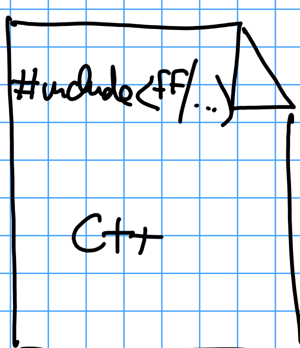
FF

pipe farm (feedback)
SPMC MPSC MPMC
code SPSC lock free / fence free
POSIX THREAD

SPSC ← consumatore

↑ produttore

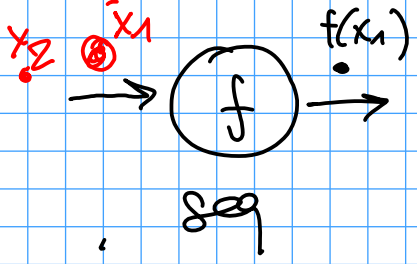




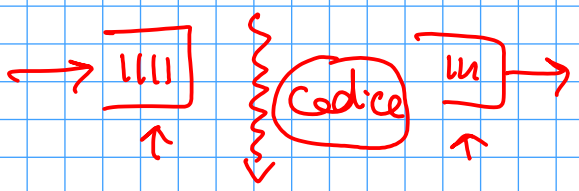
p.cpp

→ POSIX

g++ -I .... -O3 p.cpp



classe ff_node



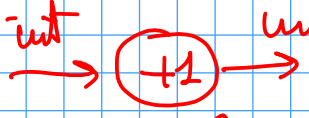

```
class f : ff_node {
    void * svc( void * );  // corpo della mia attività concorrente
};
```

task in ingresso

risultato in uscita

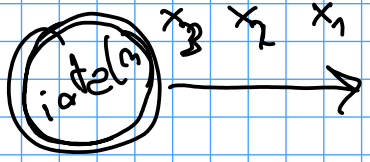
```
int svc_init(void);

void svc_end(void *);
```

```
class inc : ff_node {
    void * svc(void * t) {
        int * x = (int *) t;
        (*x)++;
        return(t);
    }
};
```



∀ elemento in ingresso provoca una chiamata alla svc

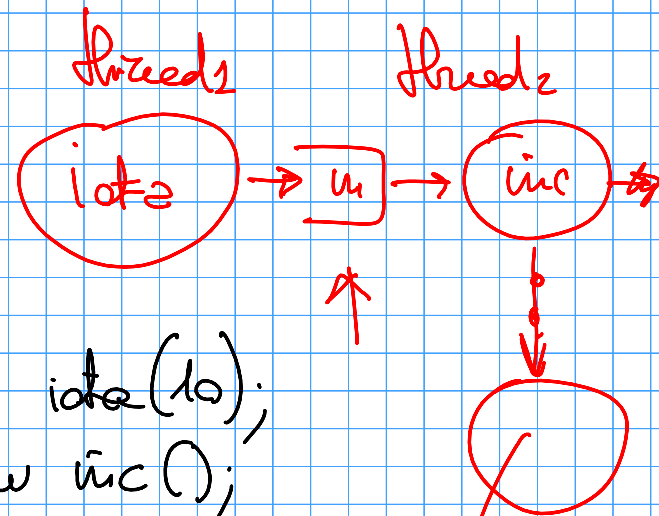
∀ return dalla svc provoca la spedizione di un risultato



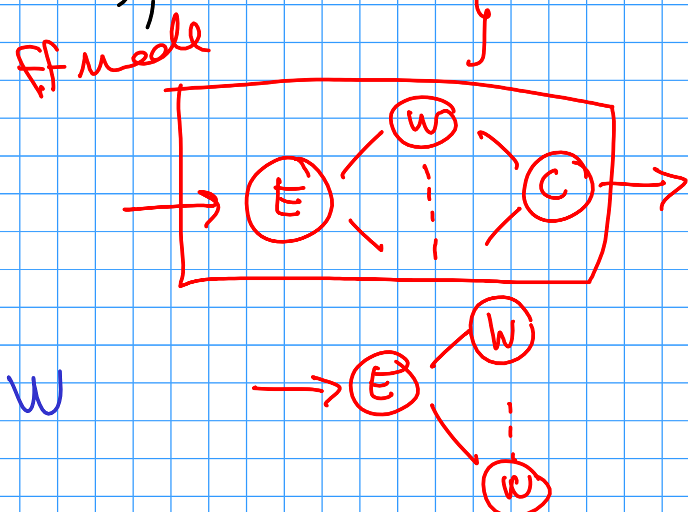
ff_send_out(void *)

```
class iota: ff_node {
  private:
    int n;
  public:
    iota(int n):n(n);

    void * svc(void * t) {
      for(int i=0; i<n; i++) {
        int * x = (int *) calloc(1, sizeof(int));
        *x = i;
        ff_send_out((void *) x);
      }
      ff_send_out(ff_eos); // ff_send_out(NULL);
    }
```

```
#include <ff/pipeline.hpp>
#include <ff/ff_node.hpp>

main() {

  ff_node primo_stadio = new iota(10);
  ff_node secondo_stadio = new inc();

  ff_pipe main;
  main.add_stage(primo_stadio);
  main.add_stage(secondo_stadio);

  main.run_and_wait_end();
```

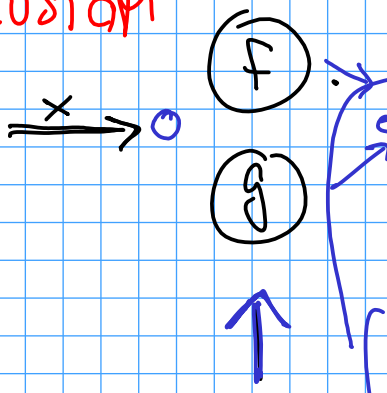



```
svc( ) {
  ...
  return(GO_ON)
}
```




```
main( ) {

  vector<ff_node> W
    for( N )
      W.push_back(new inc());

  ff_farm main;
    main.add_workers(W);
      add_emitter
      add_collector
```

CUSTOM



MISD

add-collector(
gatherall())

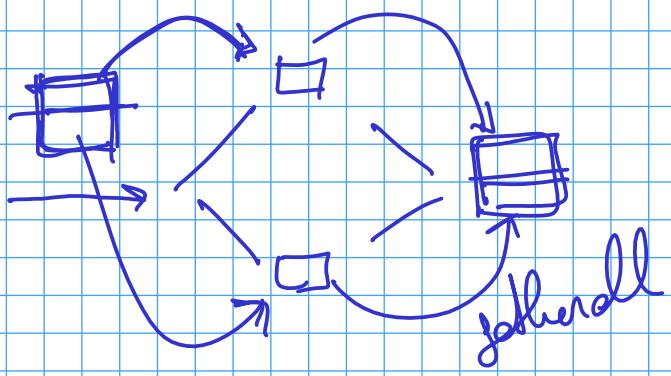
vector<ff_node> w

w.push_back(new f())

" " (new g())

farm.add_workers(w);

farm.add_emitter(broadcast());



ACCELERATORE (SW)