Athapascan-0b for PVM: adding threads to PVM

Michel Christaller

email: Michel.Christaller@imag.fr

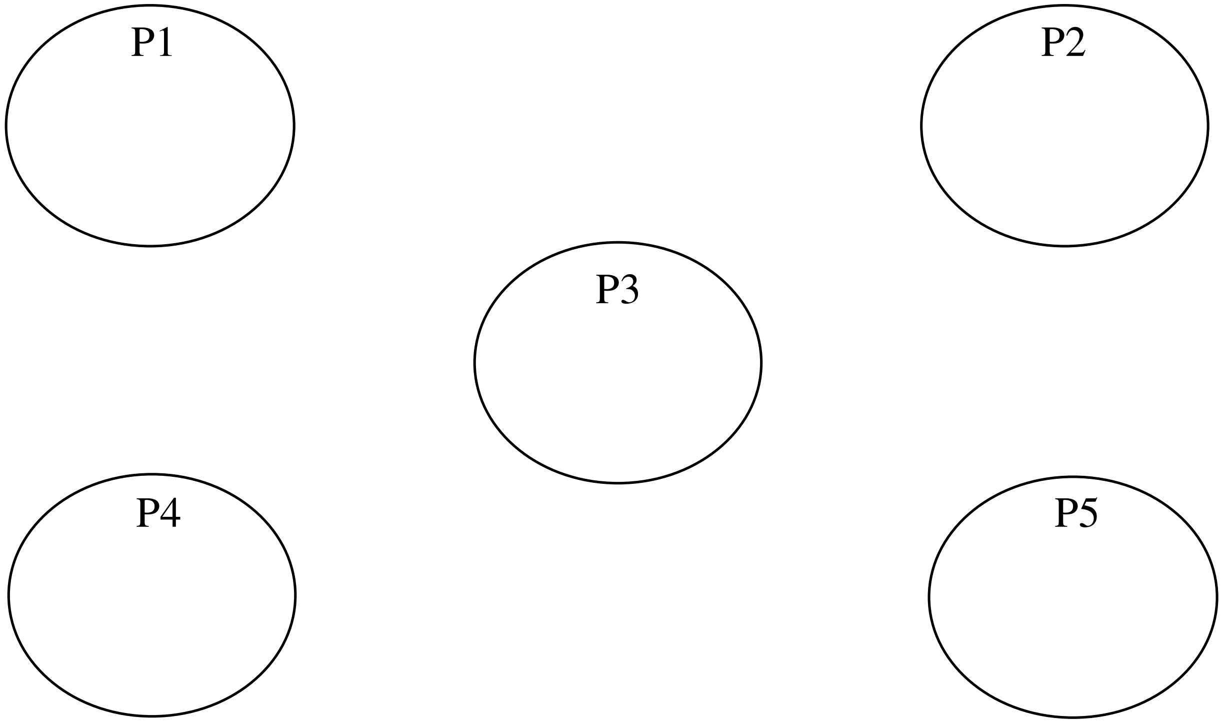
APACHE project

Institut de Mathématiques Appliquées de Grenoble

APACHE-IMAG, 46 Avenue Félix Viallet, F-38031 Grenoble Cedex, France

Need for threads

Ex: Computation of A^2xB^{-1}, A and B matrices



- Sequentially, first step

A^2 A^2

A^2

A^2

- Sequentially, second step

B^{-1} B^{-1}

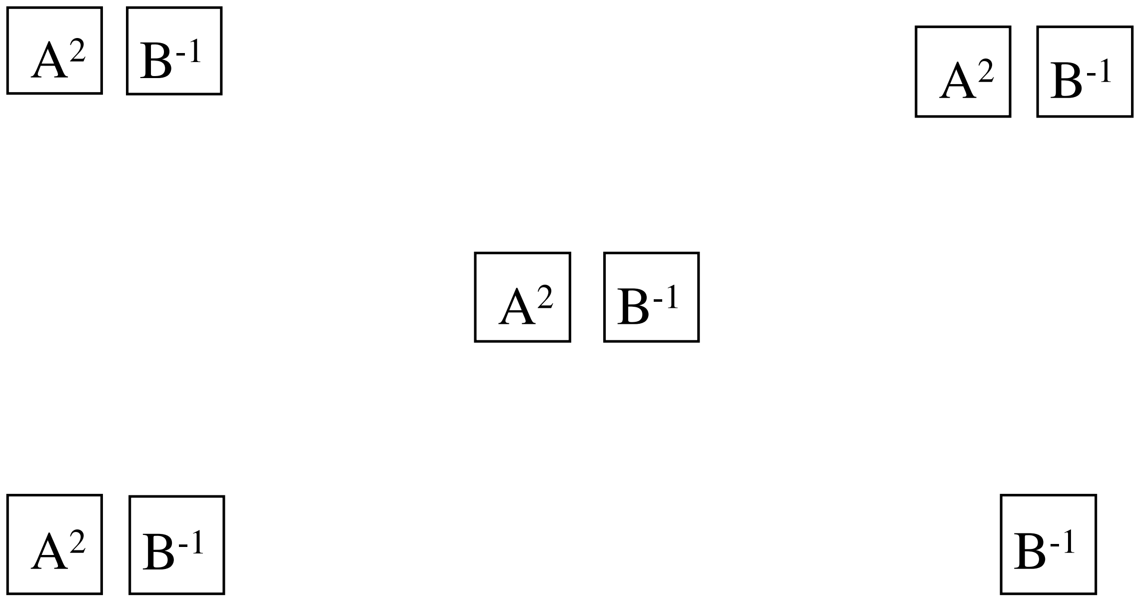
B^{-1}

B^{-1} B^{-1}

⇒ may not use all processors due to algorithm “topology”

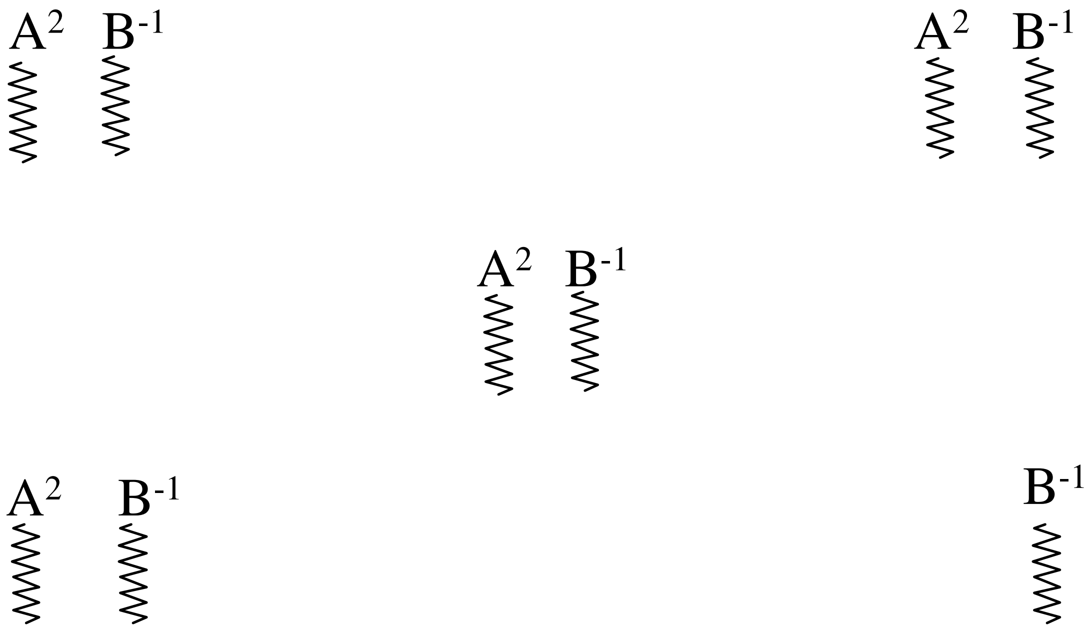
⇒ processors idle wait for messages

- In parallel, through Unix processes



⇒ costly to spawn, to switch between
⇒ unwanted protections

- In parallel, through threads



⇒ load balancing can be attempted
⇒ communication delays hidden by other threads computations
⇒ fast creation and switching: fine grain parallelism
⇒ no protection boundary to cross

PVM & Threads: Athapascan-0b for PVM

- portability
- keep efficiency
- ease of use
- foundation for other tools

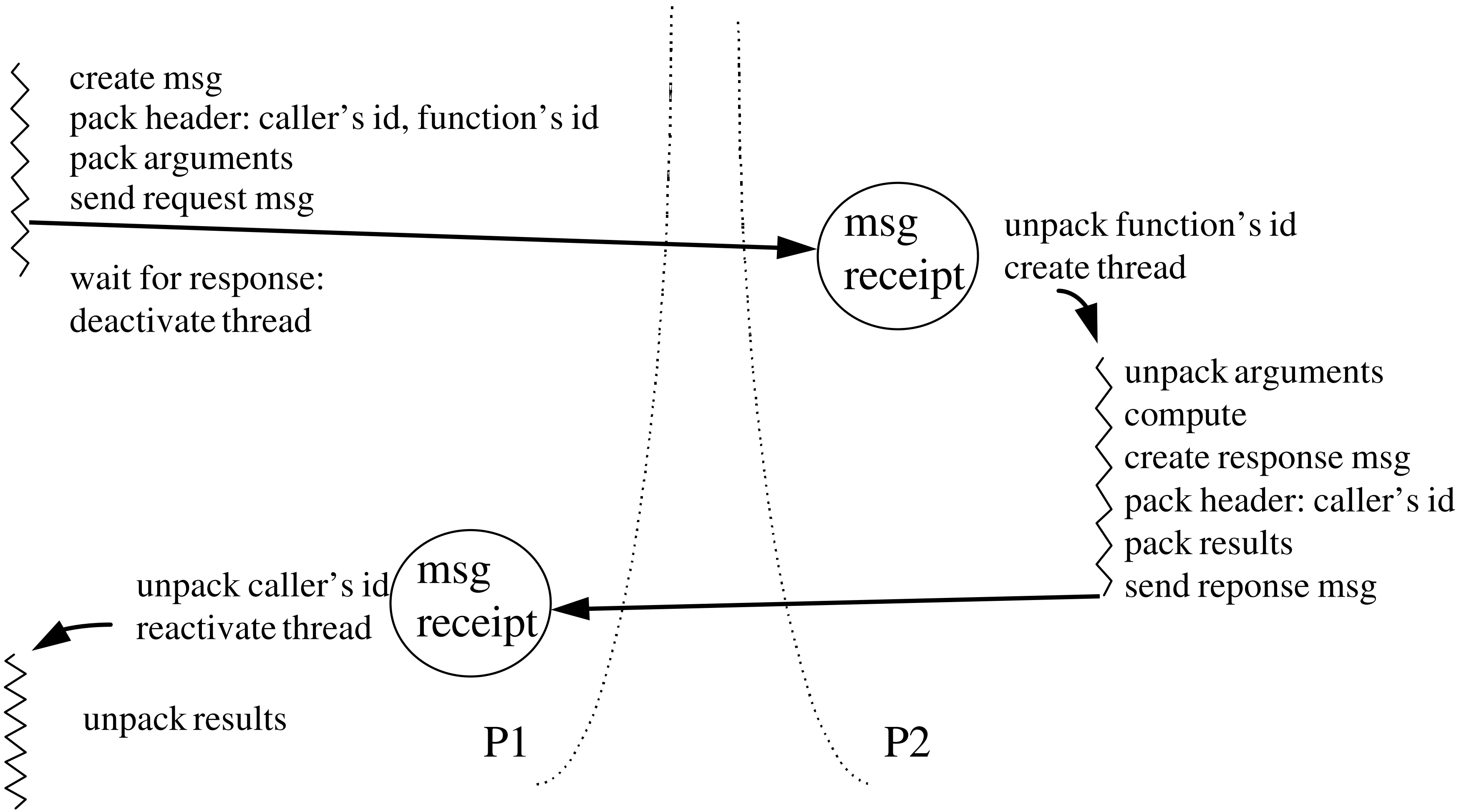
Athapascan-0b:

-remote thread creation
-thread deactivation when wait for results
-thread reactivation when receipt of results
-communication between threads

PVM
(untouched)

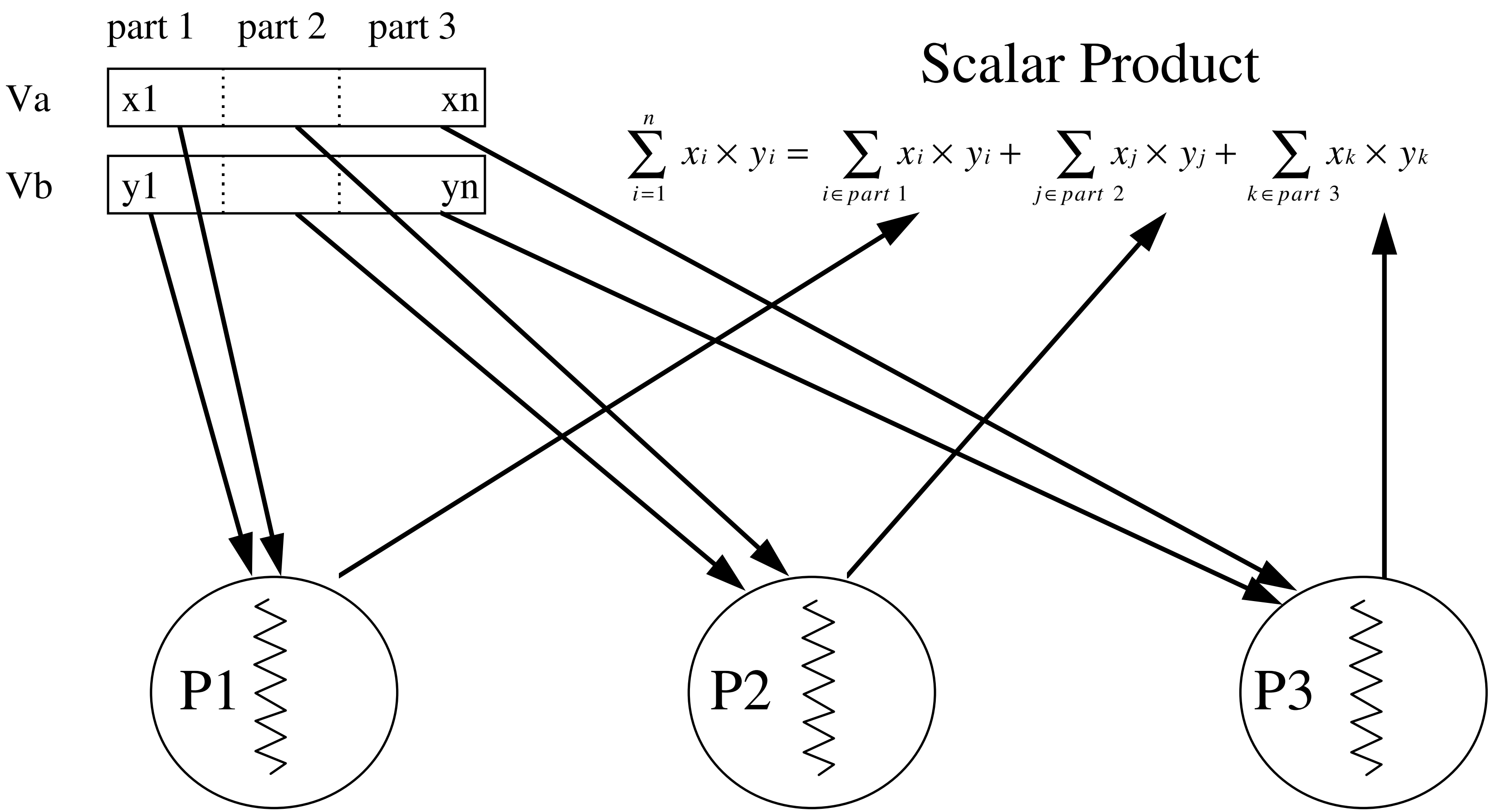
Threads Kernels:
POSIX (DCE)
LWP (Sun Os 4)
SOLARIS (Sun Os 5)
REX library (Linux PCs, ...)
(non preemptive threads)

Remote thread call protocol



- no preemption: compute as far as possible
- no interrupt driven receipt:
  - msg receipt through polling when no thread ready to run
  - no defined scheduling other than processor consistency

Example of remote thread use



Each processor computes the scalar product of both parts it gets

Scalar Product: Caller side

```
main(int argc, char ** argv) {
    int Va[n], Vb[n], r[p];
    ath_t_handle hdl[p];                          handles to remote threads
    int task[p];
    ...
    for (i=0; i<p; i++) {
        hdl[i]=ath_call(task[i], ep_scalprod);    prepare request msg
        pvm_pkint(&(Va[i*k]), k, 1);              pack arguments
        pvm_pkint(&(Vb[i*k]), k, 1);
        ath_flush(hdl[i]);                        send request
        }

    for (i=0; i<p; i++) {
        ath_wait(hdl[i]);                         wait for reponse msg
        pvm_upkint(&(r[i]), 1, 1);                unpack results
        ath_close(hdl[i]);                        free internal structures
        }

    sum all r[i]
}
```

Scalar Product: Callee side

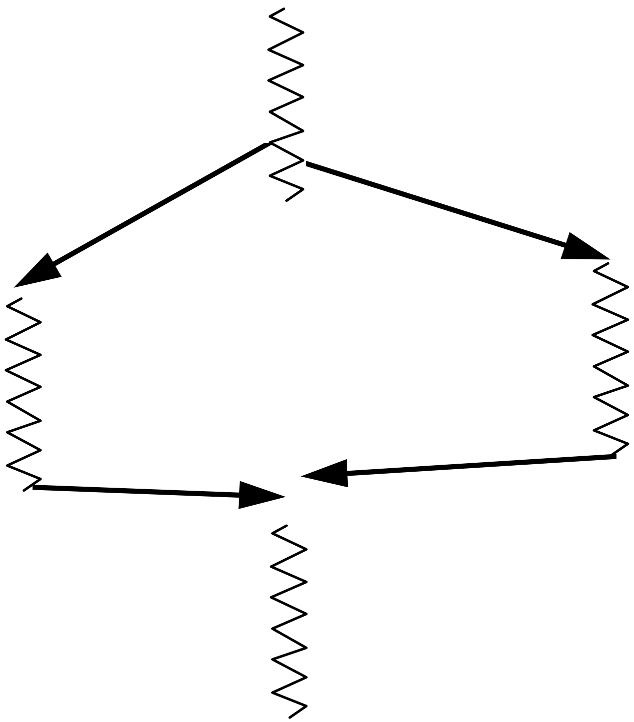
```
main(int argc, char ** argv) {

    ath_init();                                  initialize athapascan executive
    ath_newep(ep_scalprod, scalprod, null);      register exported functions
    ath_serve();                                 begin to serve requests
}

void  scalprod(ath_t_handle hdl) {
    int Wa[k], Wb[k], r;
    int i;                                       pvm receive buffer = request msg
                                                 pvm send buffer = response msg
    pvm_upkint(&Wa, k, 1);
    pvm_upkint(&Wb, k, 1);
    r=0;
    for (i=0; i<k; i+=1)
        r+=Wa[i]*Wb[i];
    pvm_pkint(&r, 1, 1);
}                                                response msg automatically sent
```

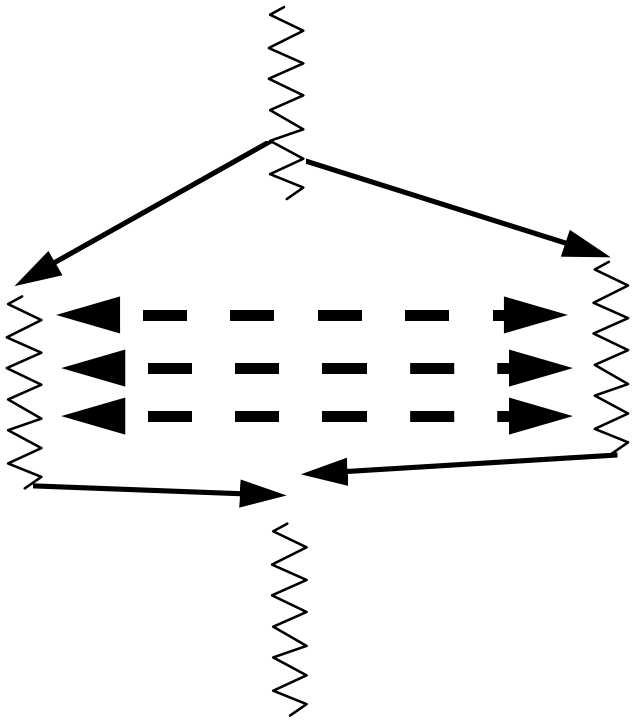
Computation models

RPC-like



independent threads

Cooperative



threads cooperating to do
a collective computation

Communication between threads

⇒ RPC like mechanism insufficient

- direct send-receive of messages between threads
  - extension of message passing model from tasks to threads
  - *maybe too low level of programming*
- through "pipes"
  - established during RPC call (father-son links)
  - established between sibling threads
  - *a mean to organize communications: use of the "topology" of the algorithm*
- through shared memory interactions
  - remote threads = remote read / write of memory
  - ADA-like accept: trap the thread creation request into current execution to control accesses to memory

Conclusion

- Evolution of Athapascan-0a:

  parallel language -> library for threads & communications
- RPC easy to implement, communications not finished
- will be available on 1st november by anonymous ftp on ftp.imag.fr (in the pub/APACHE directory)
- Performance:
  - overhead of some bytes in all messages (headers)
  - overhead of response management
  - ability to hide communication delays by computations
  - benchmarks on real applications to be done
- PVM problems:
  - impossible to modify the header after packing arguments
  - no interrupt at message arrival -> no real-time events (scheduling, load balancing informations, ..)