FAULT TOLERANCE

NF CONCERN

PGM → t

LONG RUNNING (h → d)

Snapshot of the state → CHECKPOINT

→ PC → where

{MEM} → state (what)

Δt^{-1d}

σ_1 σ_2 σ_3

t_{check}

$T_{seq} + M T_{check}$

$T_{seq} / \Delta t$

$X_1 X_2 X_3 \ldots X_m$

FAST ETHERNE NOW

PE_1 — PE_2, PE_3, PE_4, PE_5 — PE_6

read

write

$F(x_1)\ F(x_2)\ F(x_3) \ldots F(x_m)$

$PIX_{i,j}$

CLOCK SKEW

PE_1 snapshot

PE_2 snapshot

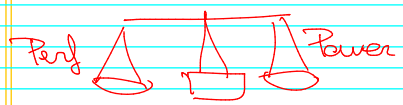
t

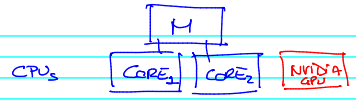
① PROTOCOLS TO AGREE ON "WHEN" WE DO CHECK POINTS

② PC + STATE + COMMS → SNAPSHOT

POWER MANAGEMENT

Mobile systems ⟷ HPC
(→ batteries) (→ cost of power)



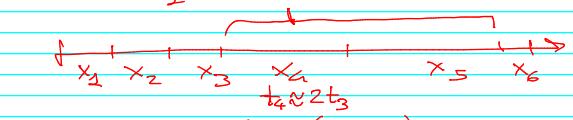


$\forall i$ $img_i \xrightarrow{f} img'_i$ CPU f GPU f

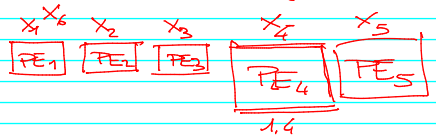
img_{di} move it to GPU(M) f on GPU

NFCODE move back img'_{di} to CPU(M)

$G = (A, N)$ $x_1 x_2 \dots x_m$ → color → BFW → $f(x_1) \dots f(x_m)$

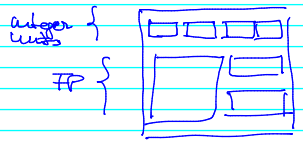


SCHEDULING (CHOICE)



RESEARCH PROJECTS

HETEROGENEOUS MULTI CORES



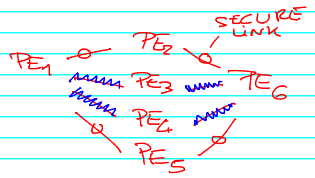
7 cores
4 small
1 bis
2 medium

SECURITY

NOW / COW — PUBLIC NETWORK SEGMENT

GRIDS

SECURE — DATA / CODE



deploy code

send/receive data

PLAIN SOCK — send

SSL SOCK — send + cypher

$PE_i \equiv$ TCP/IP POSIX

SEC → SSL sockets instead of PLAIN TCP/IP sockets

$PuKey_2(K) \longrightarrow PrivateKey_2(PubKey_2(K)) \rightarrow K$

COMP. ACT. GRAPH

↓

IMPLEMENTATION

↙ sec comms for unsecure links

N sec / faster comms for secure links

deploy(x) → secure nodes

deploy sec(x) → N secure nodes

x x'_{sec}

CLASSICAL // PROGRAMMING FRAMEWORKS

NETWORK OF PC/WS — LINUX — TCP/IP

MECHANISMS COMMUNICATIONS | MECHANISMS SYNCHRONIZATION | CONCURRENT ACTIVITIES

+

TOOLS

DEPLOYMENT. — RUNNING

CONCURRENT ACTIVITIES

PROCESSES — PTHREADS

~~fork()~~ CREATE

exec(...)

DIFFERENCIATE

pthread_create

void * F(void *)

body

P1 → P2

void * p1 (void *)
void * p2 (void *)
main
{
pthread_create(p1)
pthread_create(p2)

main
~~pid~~
fork — exec(P1) — TCP/IP
P2 — exec(P2)

COMMUNICATIONS

INSIDE PE — BETWEEN PES

SOCKET
AF_UNIX SOCKETS; SOCKETS

SHARED MEMORY — CLIENT — SERVER

serverSocket(port)
accept()

THREADS — SYSV calls — clientSocket(IP, port)
connect(port)

lock
sem — PROCESSES — fd — write / read ⟷ fd — read / write

allocate
syscall
key
ptr

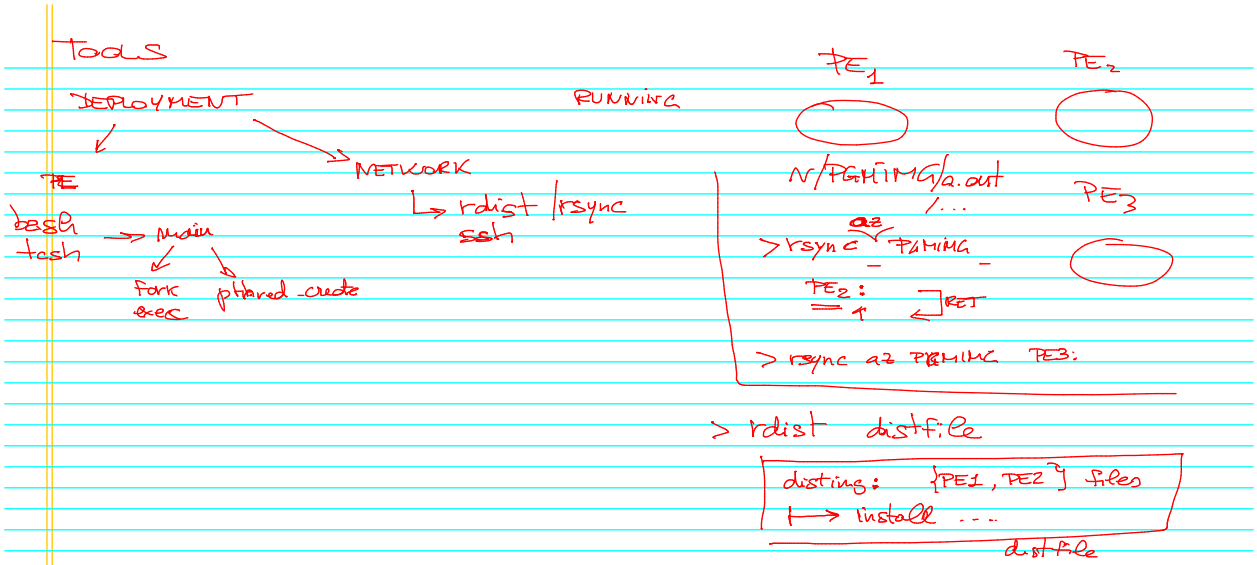
SYNCHRONIZATION

INTRA PE — INTER PE

Semaphores (SYS V – PROCESS)

THREADS
sem, lock,

SOCKETS
synchronization messages (1 byte write/read)


TOOLS
DEPLOYMENT
RUNNING
NETWORK
rdist / rsync
ssh
bash
tcsh
main
fork
exec
pthread_create
PE1
PE2
PE3
N/PGMING/a.out
rsync a2 PGMING
PE2:
RET
rsync a2 PGMING PE3:
> rdist distfile
distfile:
{PE1, PE2} files
install ...