Calculs sécurisés adaptatifs sur infrastructure de calcul global

Thierry Gautier, Samir Jafar, Franck Leprévost+,
Jean-Louis Roch, Sébastien Varrette+
and Axel Krings*

Projet MOAIS (CNRS,INPG,INRIA,UJF) LIG - IMAG, Grenoble, France
http://moais.imag.fr

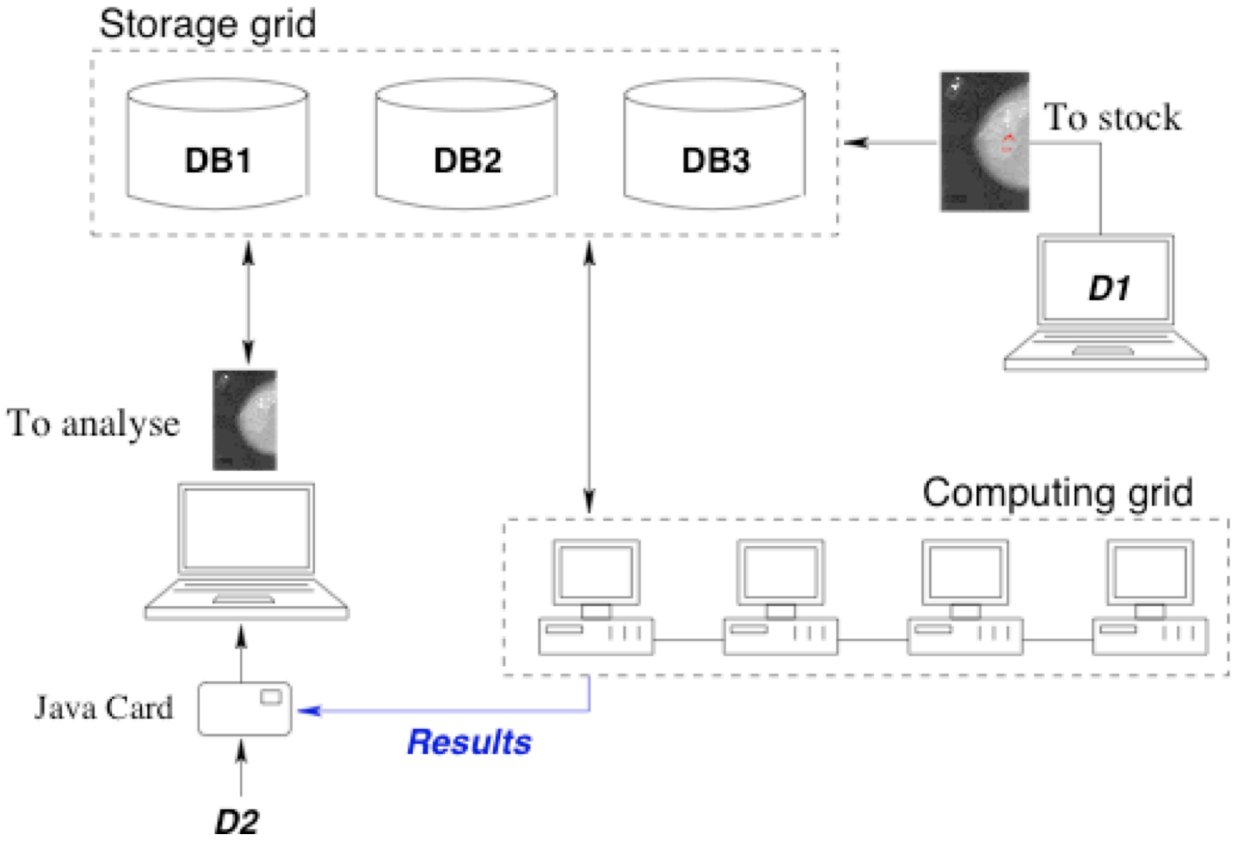
+ Université du Luxembourg, Luxembourg
* Idaho University, Moscow, Idaho .

Target Application

- Large-Scale Global Computing Systems
- Subject Application to Dependability Problems
  - Can be addressed in the design
- Subject Application to Security Problems
  - Requires solutions from the area of survivability, security, fault-tolerance

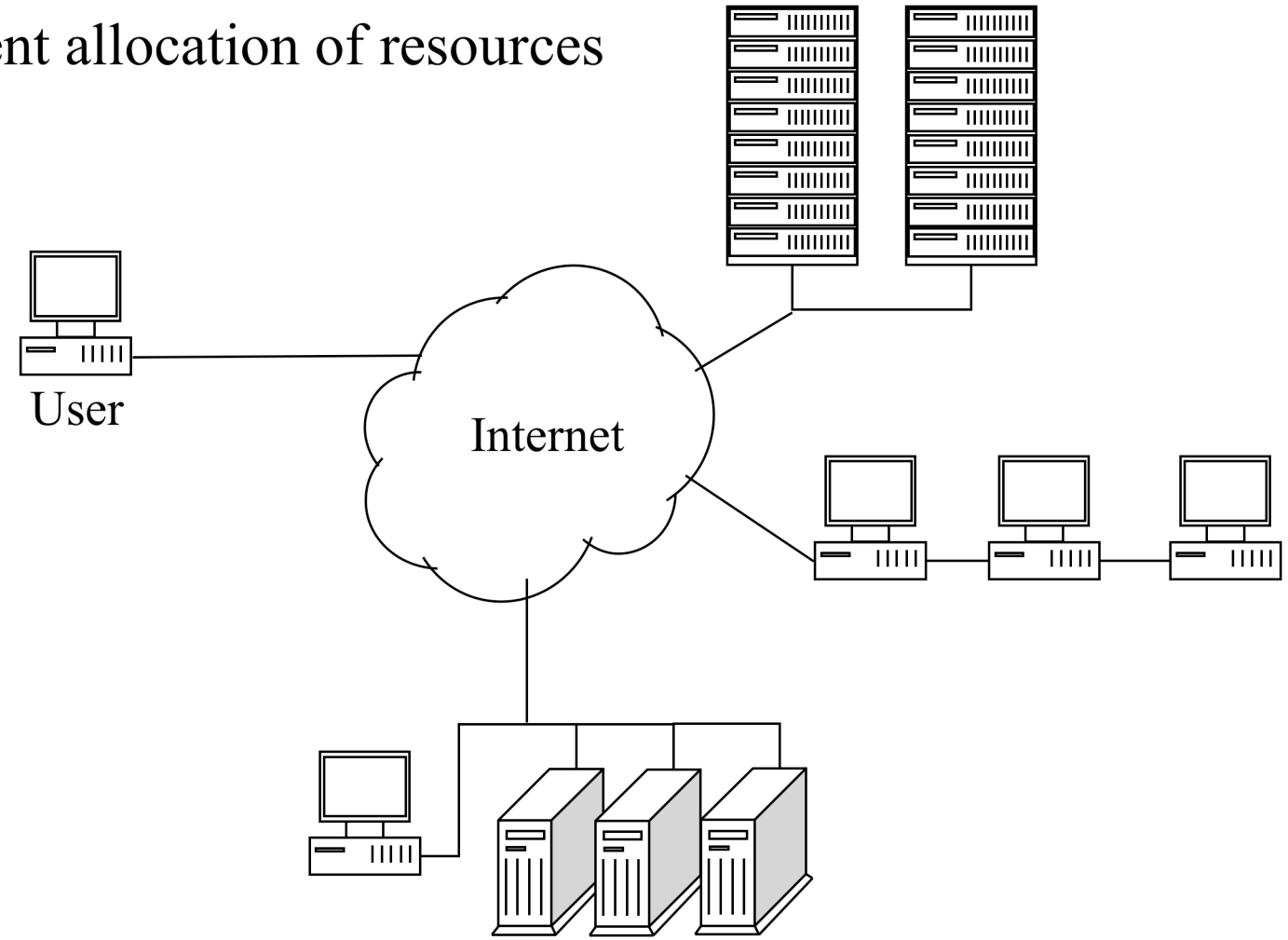
Typical Application [RAGTIME]

- Computation intensive parallel application
  - Medical (mammography comparison)



Global Computing Architecture

- Large-scale distributed systems (e.g. Grid, P2P)
  - Eg : BOINC [Berkeley Open Infrastructure for Network Computing]
- Transparent allocation of resources

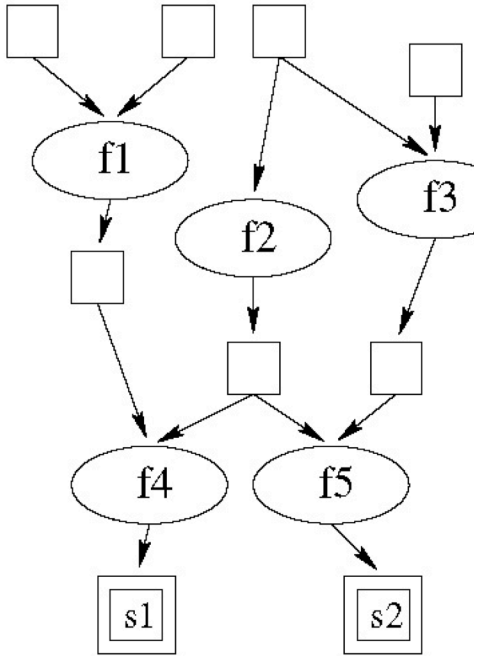


Definitions and Assumptions

- Dataflow Graph
  - $G = (\mathcal{V}, \mathcal{E})$

    $\mathcal{V}$ finite set of vertices v_i

    $\mathcal{E}$ set of edges e_{jk} vertices $v_j , v_k \in \mathcal{V}$



- Two kinds of tasks
  - T_i Tasks
    in the traditional sense
  - D_j Data tasks
    inputs and outputs

Resource allocation

- Assumption on the application :
  - large number of operations to perform = W_1 *(sequential work)*
  - huge degree of parallelism = W_∞ *(critical time = parallel work on #procs= ∞)*
  - Global computing application framework : $W_\infty << W_1$

- Allocation : Distributed randomized work-stealing schedule [Cilk98] [Athapascan98]
  - Local non-preemptive execution of tasks
    - New created tasks are pushed in a local queue.
  - When a resource becomes idle, it randomly selects another one that has ready tasks (greedy) and steals the oldest ready task

- Provable performances (with huge probability) [Bender-Rabin02]
    - On-line adaptation to the global computing platform

$$\text{Execution time} \leq \frac{W_1}{\Pi_{tot}} + \frac{W_\infty}{\Pi_{ave}}$$

Security issues for a global computation

- In the Survivability Community our general computing environment is referred to as

Unbounded Environment

  - Lack of physical / logical bound
  - Lack of global administrative view of the system.

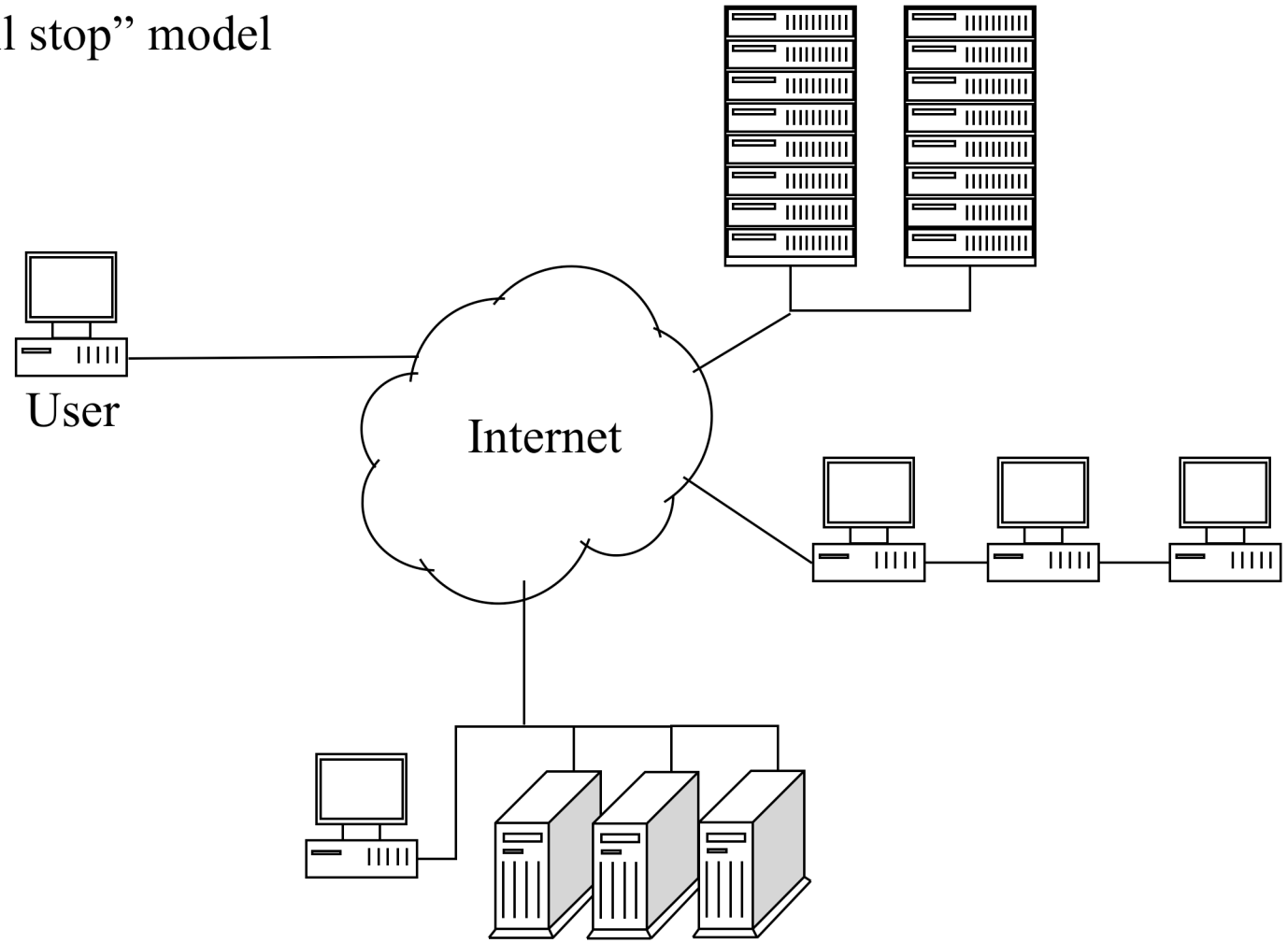
What risks are we subjecting our applications to?

Assumptions

- Anything is possible!
  - » and it <u>will</u> happen!
- Malicious act will occur sooner or later
- It is hard or impossible to predict the behavior of an attack

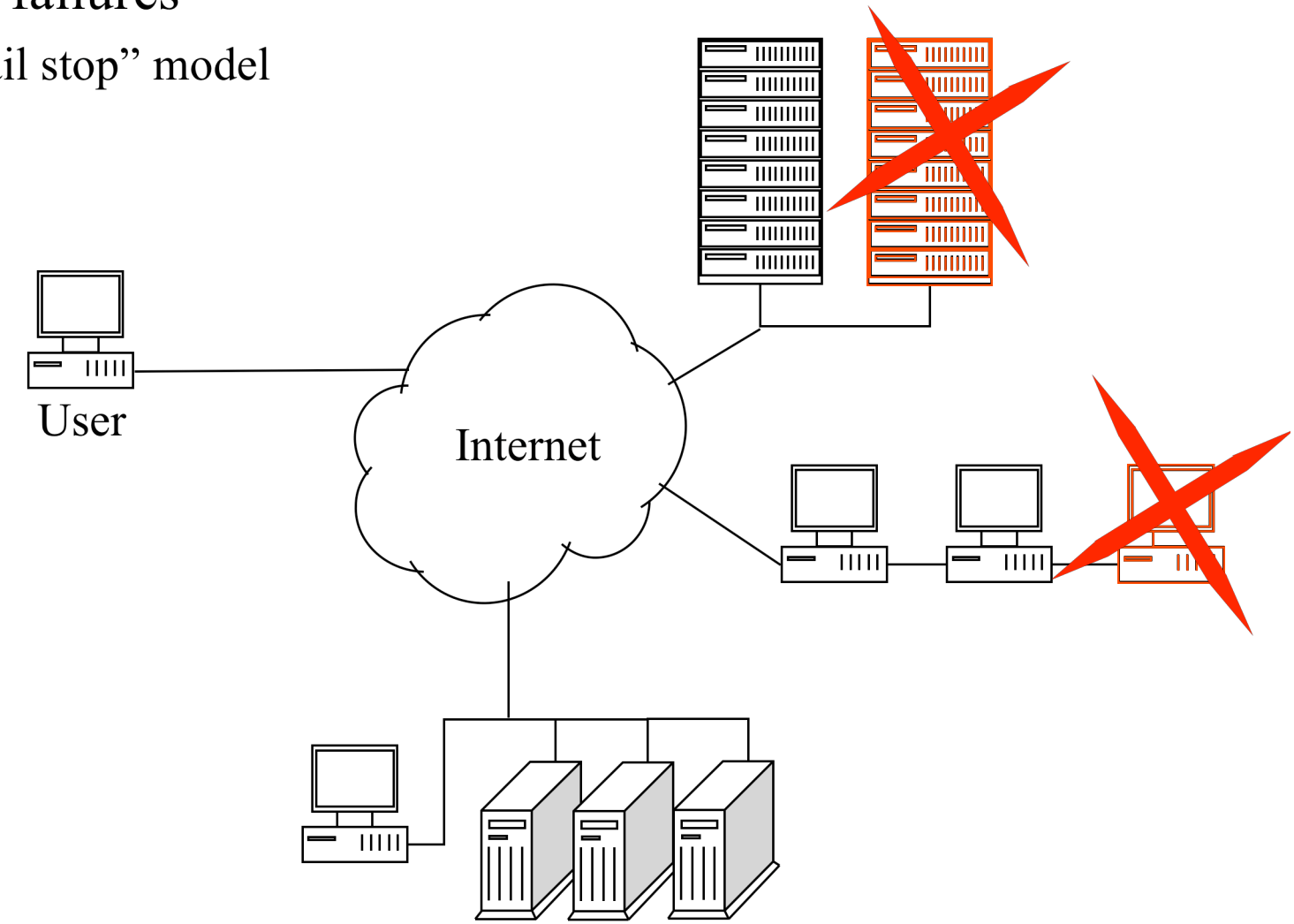
Two kinds of failures (1/2)

1. Node failures
   - "fail stop" model



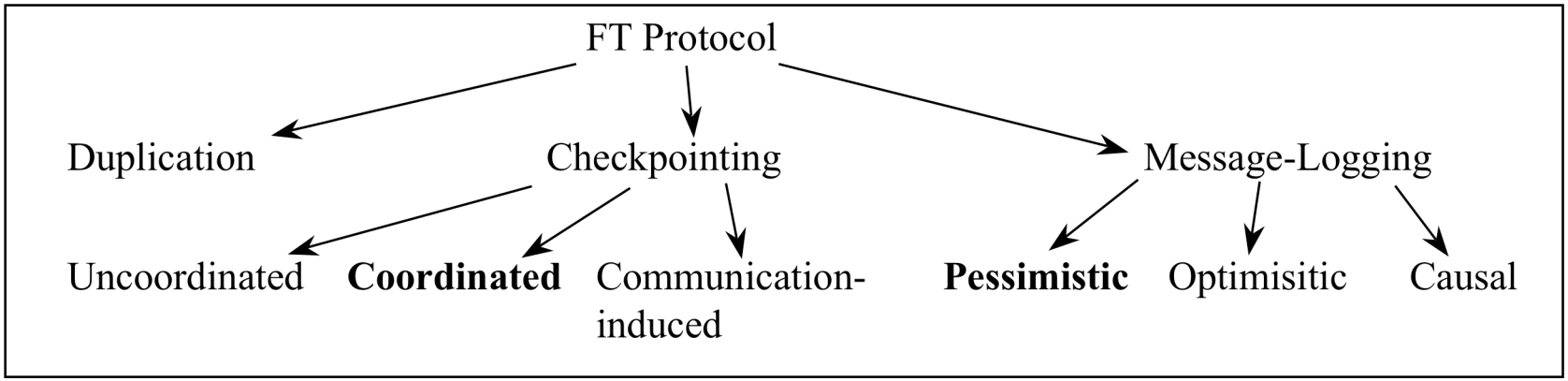
Two kinds of failures (1/2)

1. Node failures
   - "fail stop" model



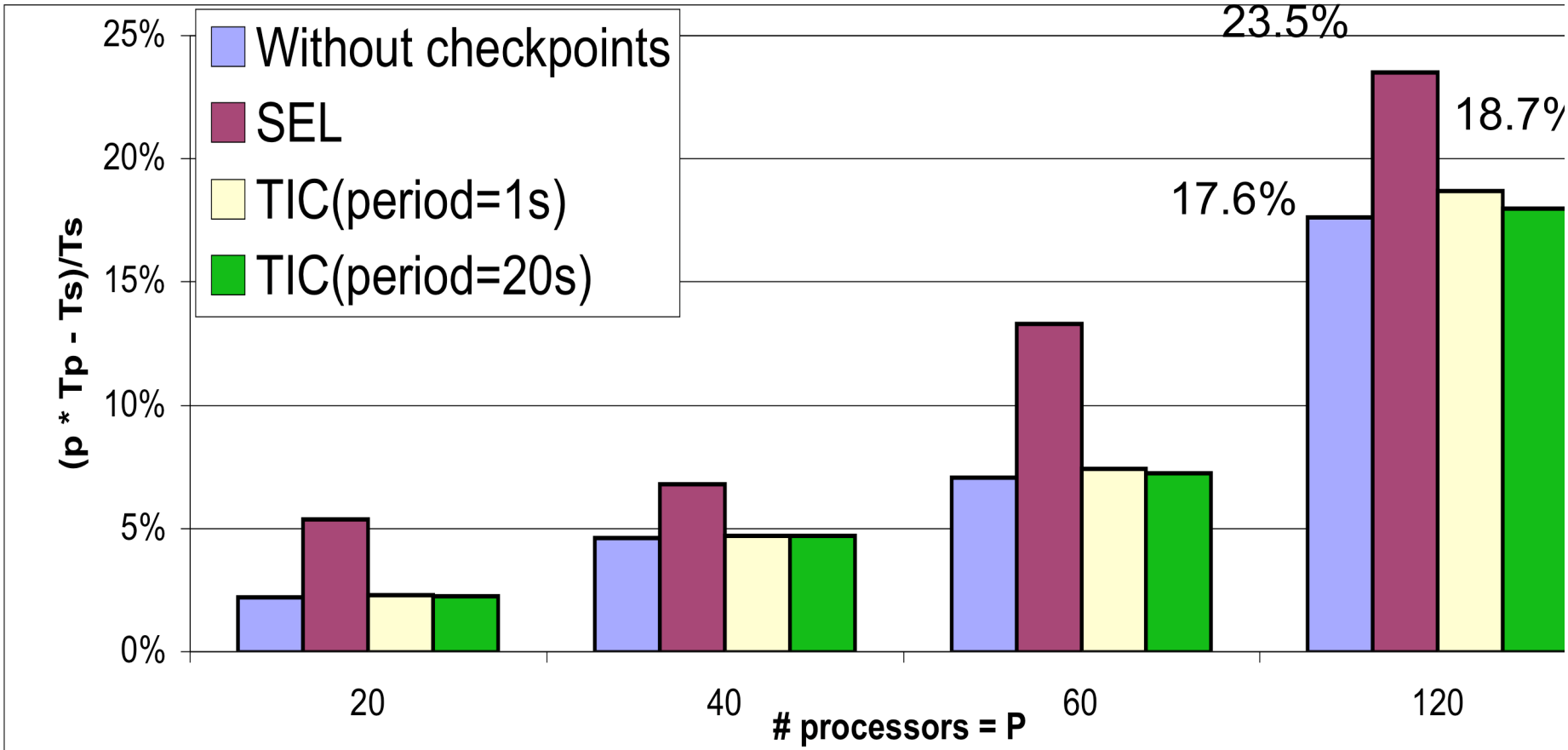
Fault Tolerance Approaches

- Simplified Taxonomy for Fault Tolerance Protocols



- Stable memory to store checkpoints (replication, ECC, ..)
- Two "extreme" protocols (**distributed**, asynchronous) are distinguished :
  - Pessimistic : Systematic storage of all events / communications :
    - Large overhead but ensures small restart time [MPICH-V1]
  - Optimistic : only events that ensure causality relations are stored [Com. induced]
    - Overhead is reduced but more recomputations in case of fault [Satin 05]
- Compromises :
  - Non-coordinated : periodic local checkpoint of the tasks queue
  - Coordinated : global checkpoint of the stacks

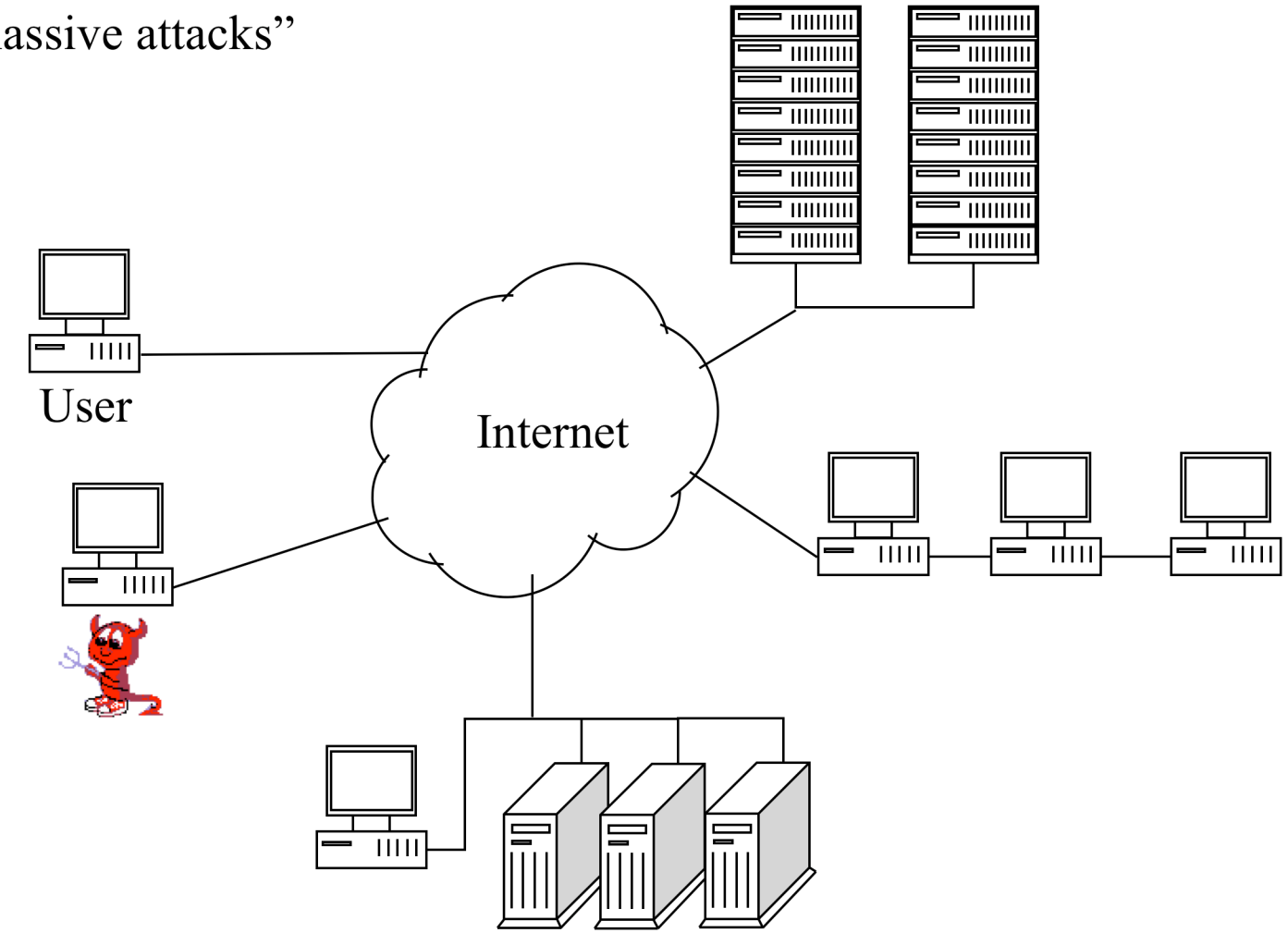
Pessimistic [SEL] storage versus non-coordinated com. induced [TIC]



Application : Quadratic Assignment Problem with Kaapi : QAP-Nugent 24 [Cung&al 05]

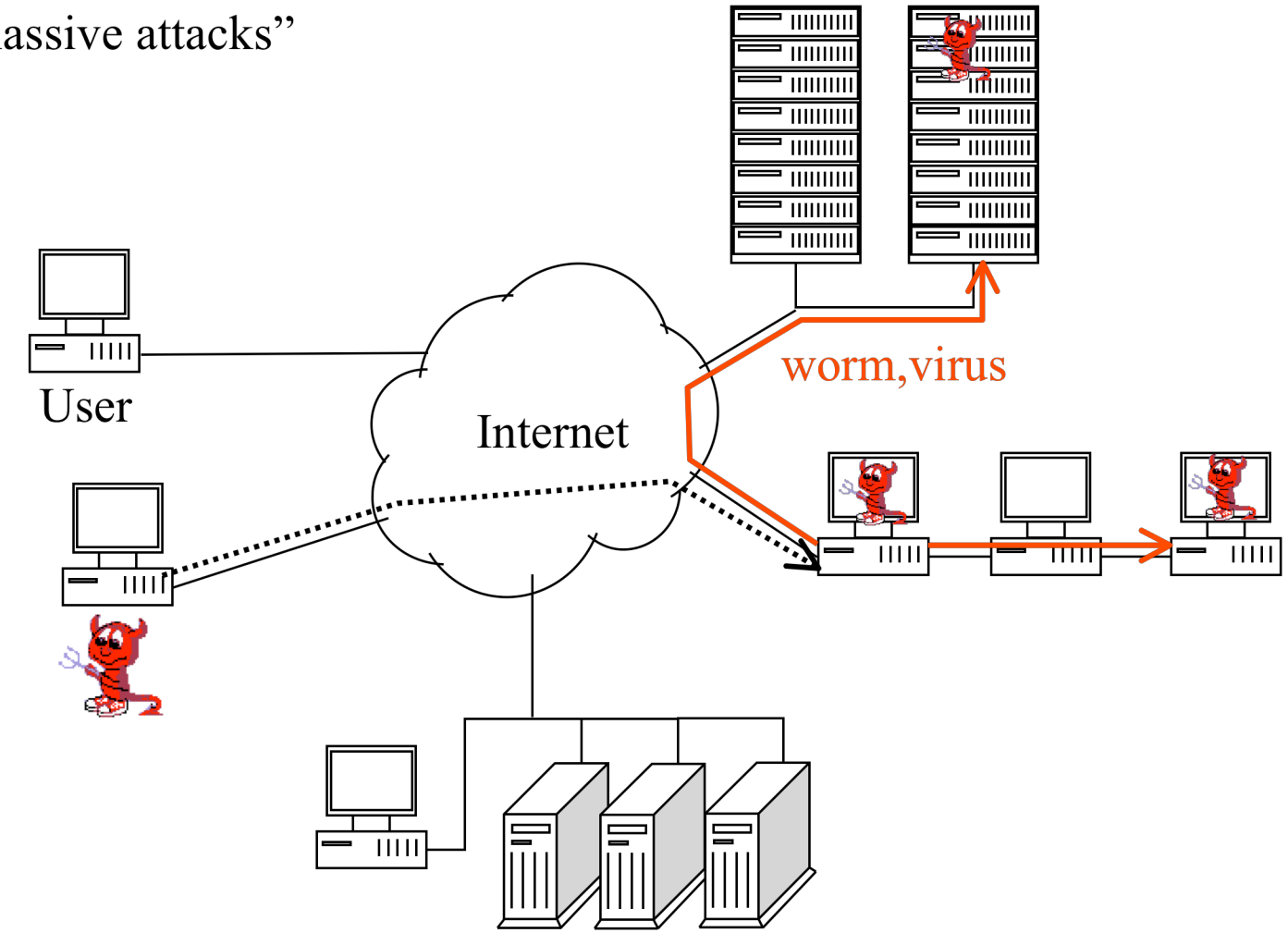
Two kinds of failures (2/2)

2. Task forgery
   - "massive attacks"



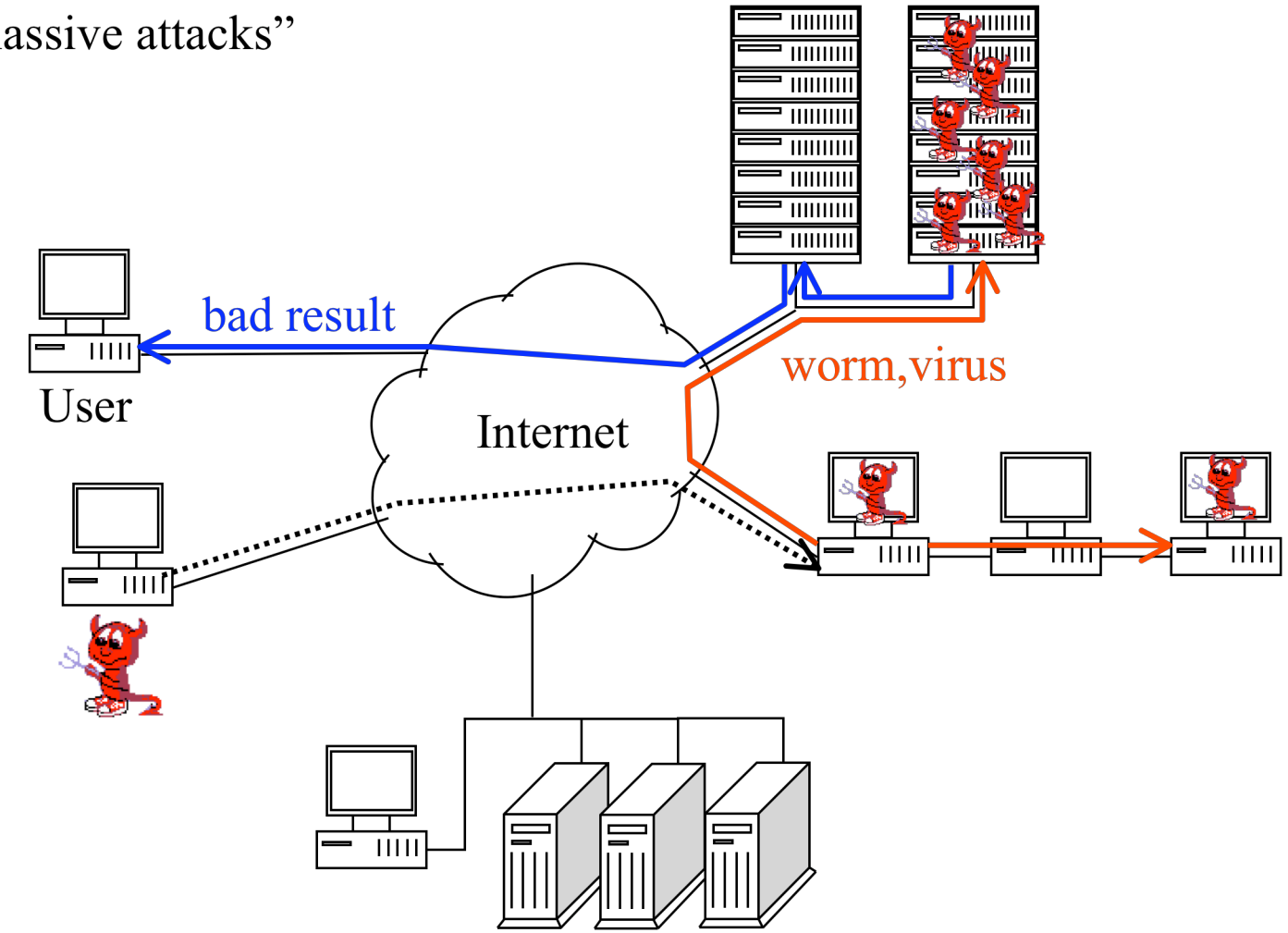
Two kinds of failures (2/2)

2. Task forgery
   - "massive attacks"



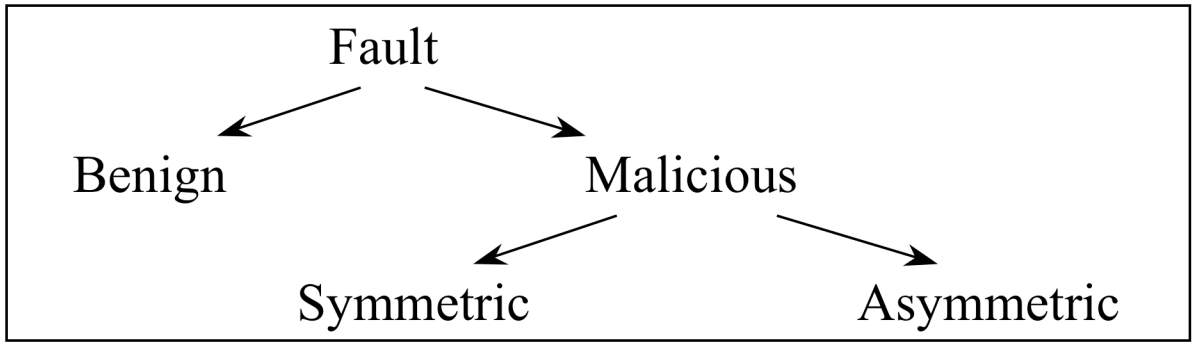
Two kinds of failures (2/2)

2. Task forgery
   - "massive attacks"



Fault Models

- Simplified Fault Taxonomy



- Fault-Behavior and Assumptions
  - Independence of faults
  - Common mode faults -> towards arbitrary faults!

- Fault Sources
  - Trojan, virus, DOS, etc.
  - How do faults affect the overall system?

Attacks and their impact

- Attacks
  - single nodes, difficult to solve with certification strategies
  - solutions: e.g. intrusion detection systems (IDS)
- Massive Attacks
  - affects large number of nodes
  - may spread fast (worm, virus)
  - may be coordinated (Trojan)
- Impact of Attacks
  - attacks are likely to be widespread within neighborhood, e.g. subnet
- Our focus: massive attacks
  - virus, trojan, DoS, etc.

Certification Against Attacks

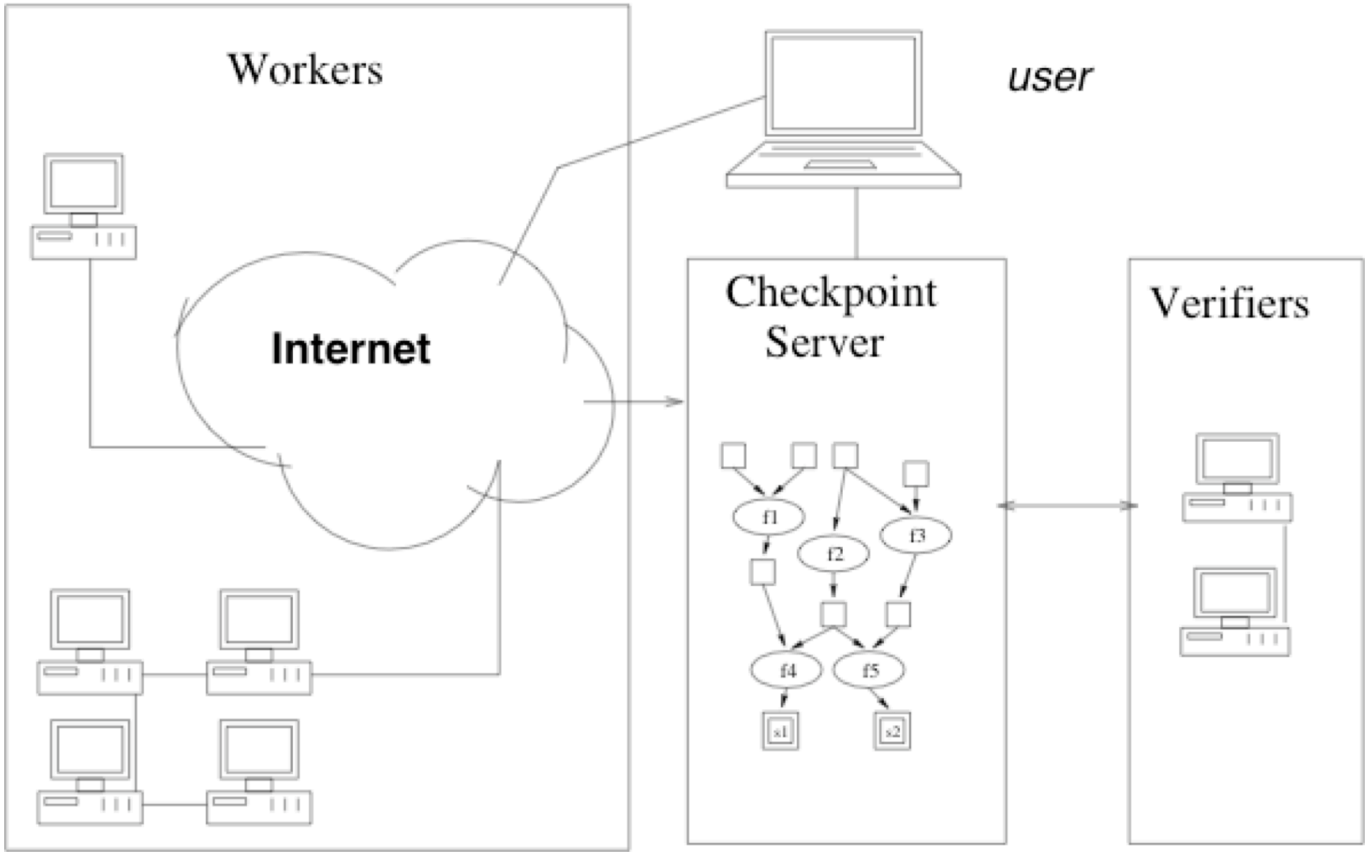
- Mainly addressed for **independent tasks**

- Current approaches
  - Simple checker [Blum97]
  - Voting [eg BOINC, SETI@home]
  - Spot-checking [Germain-Playez 2003, based on Wald test]
  - Blacklisting
  - Credibility-based fault-tolerance [Sarmenta 2003]
  - Partial execution on reliable resources (partitioning) [Gao-Malewicz 2004]
  - Re-execution on reliable resources

- Certification of Computation to detect massive attacks

Global Computing Platform (GCP)

- GCP includes workers, checkpoint server and verifiers



Probabilistic Certification

- Monte Carlo certification:
  - a randomized algorithm that
    1. takes as input E and an arbitrary ε, $0 < \varepsilon \leq 1$
    2. delivers
       - either CORRECT
       - or FAILED, together with a proof that E has failed
  - certification is with error ε if the probability of answer CORRECT, when E has actually failed, is less than or equal to ε.
- Interest
  - ε : fixed by the user (tunable certification)
  - Number of executions by the verifiers is not to large with respect of the number of tasks

Protocols MCT and EMCTs

- The Basic Protocol: The Monte Carlo Test (MCT) [SBAC04]

  1. Uniformly select one task T in G
     we know input $i(T,E)$ and output $o(T,E)$ of T from checkpoint server
  2. Re-execute T on verifier, using $i(T,E)$ as inputs, to get output $\hat{o}(T,E)$
     If $o(T,E) \neq \hat{o}(T,E)$ return FAILED
  3. Return CORRECT

- Results about extended MCT (EMCTs) [EIT-b 2005]
  - Number N of re-execution depends

$$N \geq \left\lceil \frac{\log \varepsilon}{\log(1 - \Lambda_G)} \right\rceil$$

  - where Λ_G depends on the graph structure, the ratio of tasks forgeries and of the protocol
  - E.g.: For massive attack and independent tasks: $\Lambda_G = q$

Certification of Independent Tasks

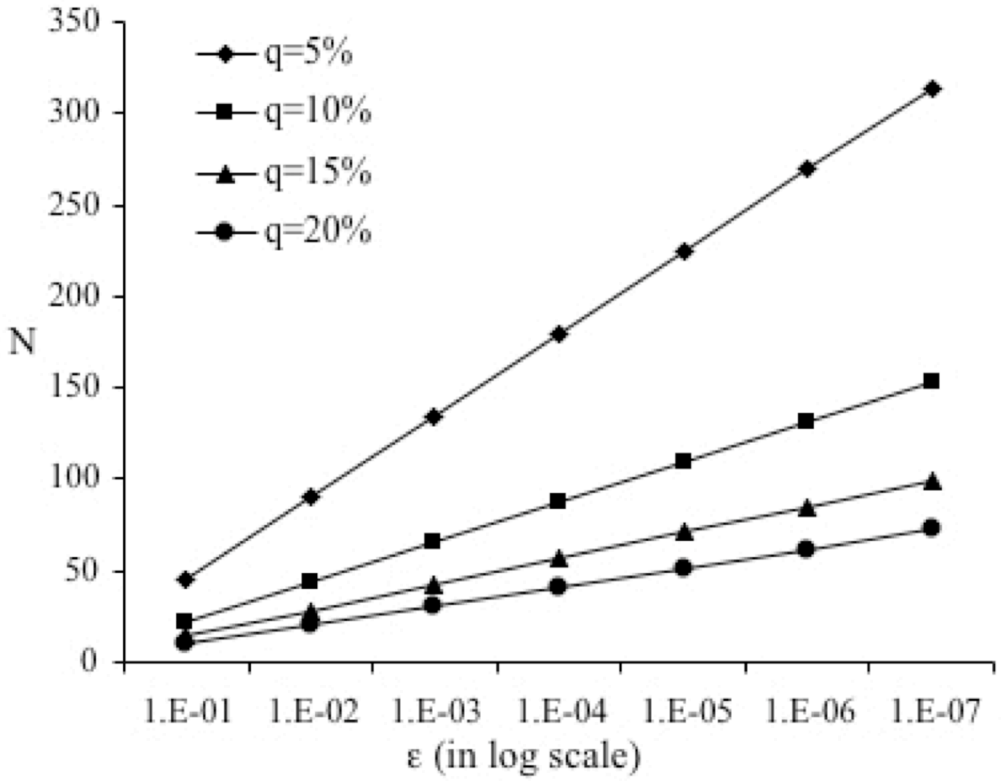
- How many independent executions of MCT are necessary to achieve certification of E with probability of error $\leq \varepsilon$?

$$N \geq \left\lceil \frac{\log \varepsilon}{\log(1-q)} \right\rceil$$

  - Prob. that MCT selects a non-forged tasks is $\frac{n - n_F}{n} \leq 1 - q$
  - N independent applications of MCT results in $\varepsilon \leq (1 - q)^N$

Certification of Independent Tasks

- Relationship between certification error and N



For q = 1% :

•300 checks => $\varepsilon < 5\%$

•4611 checks => $\varepsilon < 10^{-20}$

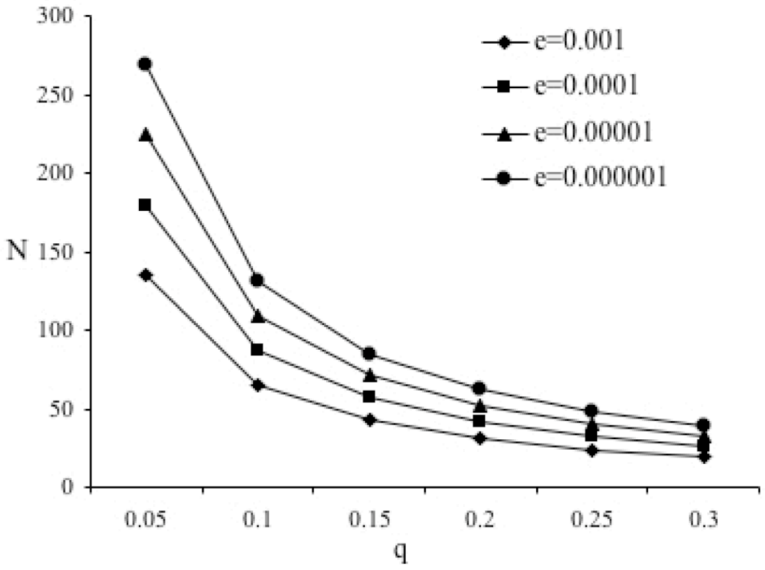
•24000 checks => $\varepsilon < 10^{-}$

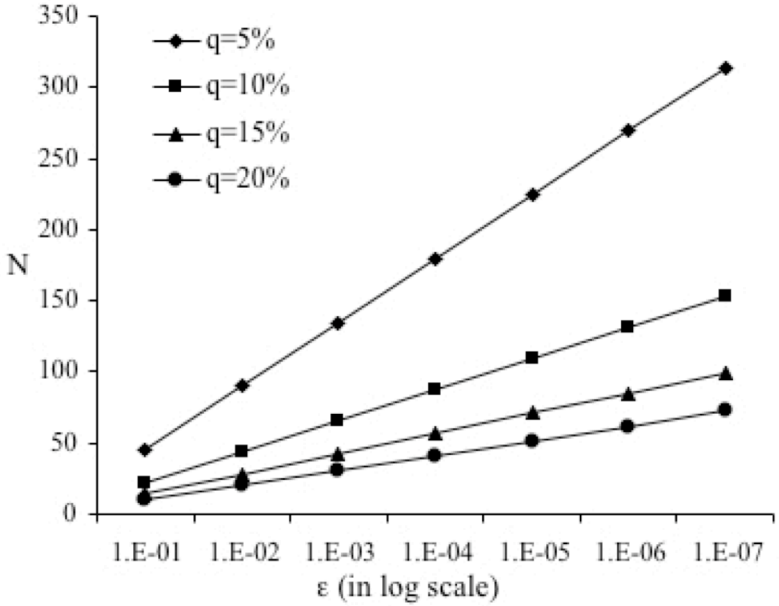
Task dependencies

- Algorithm EMCT
  1. Uniformly select one task T in G
  2. Re-execute all T_j in $G^{\leq}(T)$, which have not been verified yet, with input $i(T,E)$ on a verifier and return FAILED if for any T_j we have $o(T_j,E) \neq \hat{o}(T_j,E)$
  3. Return CORRECT
- Behavior
  - disadvantage: the entire predecessor graph needs to be re-executed
  - however: the cost depends on the graph
    - luckily our application graphs are mainly trees

Analysis of EMCT

- Results of independent tasks still hold,
  - but *N* hides the cost of verification
    - » independent tasks: C = 1
    - » dependent tasks: C = $|G^{\leq}(T)|$





Reducing the cost of verification

For EMCT the entire predecessor graph had to be verified
To reduce verification cost two approaches are considered next:

1. Verification with fractions of $G^{\leq}(T)$
2. Verification with fixed number of tasks in $G^{\leq}(T)$

Results for pathological cases

- Number of effective initiators
  - this is the # of initiators as perceived by the algorithm
  - e.g. for EMCT an initiator in $G^{\leq}(T)$ is <u>always</u> found, if it exists

	$MCT(E)$ [7]	$EMCT(E)$ [7]	$EMCT_{\alpha}(E)$	$EMCT^{1}(E$
# of effective initiators	$\lceil \frac{n_q}{\left(\frac{1-d^h}{1-d}\right)} \rceil$	n_q	$n_q \alpha \Gamma_T(n_q)$ or n_q	$n_q \Gamma_T(n_q)$
Probability of error	$1 - \frac{\lceil \frac{n_q}{\left(\frac{1-d^h}{1-d}\right)} \rceil}{n}$	$1-q$	$1 - q\alpha\Gamma_T(n_q)$ or $1-q$	$1 - q\Gamma_T(n_q$
A priori convergence	$\frac{\log \epsilon}{\log(1 - \frac{\lceil \frac{n_q}{\left(\frac{1-d^h}{1-d}\right)} \rceil}{n})}$	$\frac{\log \epsilon}{\log(1-q)}$	$\frac{\log \epsilon}{\log(1-q\alpha\Gamma_G(n_q))}$ or $\frac{\log \epsilon}{\log(1-q)}$	$\frac{\log \epsilon}{\log(1-q\Gamma_G(n_q$
q_e a priori	$\frac{\lceil \frac{n_q}{\left(\frac{1-d^h}{1-d}\right)} \rceil}{n}$	q	$q\alpha\Gamma_G(n_q)$ or q	$q\Gamma_G(n_q)$
q_e run-time	$\frac{\lceil \frac{n_q}{\left(\frac{1-d^h}{1-d}\right)} \rceil}{n}$	q	$q\alpha\Gamma_T(n_q)$ or q	$q\Gamma_T(n_q)$
Verification cost (exact)	1	$\lvert G^{\leq}(T) \rvert$	$\lceil \alpha \lvert G^{\leq}(T) \rvert \rceil$	1
Max. cost (out-tree)	1	h	αh	1

- Efficient massive attack detection in the framework $\boldsymbol{W_{\infty} << W_1}$

Conclusion

- Programming an application on a Global computing platform :
  - Designing **adaptive algorithm** for efficient resource allocation
- Managing resource resilience and crash faults :
  - **Tuned fault-tolerance** protocol to decrease overhead
  - Key problem : efficient distributed stable memory [ECC promising]
- Managing malicious intrusions :
  - Detection of massive attacks
    - Efficient **probabilistic** certification
  - Protection against local attacks :
    - Redundant computations
    - Self fault-tolerant algorithms [eg Lamport sorting network] [Varrette06]

Questions?

http://www-id.imag.fr/Laboratoire/Membres/Roch_Jean-Louis/perso_html/publications.html

[89] Samir Jafar, Varrette Sébastien, and Jean-Louis Roch. Using data-flow analysis for resilience and result checking in peer-to-peer computations. In IEEE DEXA'2004, Zaragoza, August 2004.

[92] Sébastien Varrette, Jean-Louis Roch, and Franck Leprévost. Flowcert: Probabilistic certification for peer-to-peer computations. IEEE SBAC-PAD 2004, pages 108-115, Foz do Iguacu, Brazil, October 2004.

[97] Axel W. Krings, Jean-Louis Roch, and Samir Jafar. Certification of large distributed computations with task dependencies in hostile environments. IEEE EIT 2005, Lincoln, May 2005.

[99] Samir Jafar, Thierry Gautier, Axel W. Krings, and Jean-Louis Roch. A checkpoint/recovery model for heterogeneous dataflow computations using work-stealing. EUROPAR'2005, Lisbonne, August 2005.

[104] J.L Roch & AHA Team. Adaptive algorithms : theory and application. SIAM Parallel Processing 2006, San Francisc, February 2006