From Abstract Distributed Model Checking to Concrete Implementation

Christophe Joubert

PDMC'03

July 14, 2003

INRIA Rhône-Alpes / VASY

1. Introduction

Introduction: PDMC problems

- Task partitioning:
  shared vs. distributed memory, multithreaded, …
- Load balancing:
  dynamic vs. static, distributed disk-based, …
- (Canonical) state and graph representation:
  explicit vs. implicit (BDD), game graphs, BES, XDR, compaction, …
- Termination detection:
  tree vs. ring, wave vs. acyclic, symmetric vs. central, …

Introduction: Communication

- Communication problem
  - Low-overhead communication
  - Maintaining a good proportion between computation at each process and communication
- Usually, communication is not a bottleneck, but it
  - affects all PDMC distributed memory computations, depending on different orderings and communication mechanisms used
  - is traditionally experimented on small parallel architecture (<64 nodes), hiding possible scalability issues of existing solutions
- Automatic mechanisms to solve it
  - but pitfalls (resource limits, scalability, performance, …)
  - Communication layer not clearly described

Outline of the talk

1. Introduction
2. Message Passing mechanisms
3. Distributed Model Checking (DMC) communication
4. Communication paradigms
5. Conclusion

2. Message passing mechanisms

Message passing: Strengths

- Aggregate power and memory of many computers (massively parallel architectures):
  - Clusters of cheap PCs
  - Loosely-connected environments of workstations

- 3 widely used mechanisms:
  - TCP/UDP sockets over IP
  - PVM and MPI
  - RPC and Active Message

Message passing: Weaknesses

- Low-overhead message passing is critical for performance:
  - Latency
  - Thread management
  - Data copying
  - Data buffering
  - Computation overlapping

Some message passing mechanisms present more avoidable communication overhead for DMC than other,

➔Which one is the most appropriate to DMC ?

Outline

1. Introduction
2. Message Passing mechanisms
3. DMC communication
4. Communication paradigms
5. Conclusion

3. DMC communication

DMC communication: Example

- Distributed state space generation
  - 3 main interleaved activities:
    - SEND
    - UPDATE
    - RECV
  - Overlapping
    - asynchronous sequential
    - multithreads ([])
    - ➔deadlocks and overhead

$V_i := \emptyset;\ E_i := \emptyset;\ T_i := \emptyset$

if $h(x_0) = i$ **then** $V_i := \{x_0\}$ **endif**

while $\left(\bigcup_{i=1}^{n} V_i \neq \phi\right) \vee \left(\bigcup_{i=1}^{n} channels_i \neq \phi\right)$

  if $\exists x \in V_i$ **then**

    $V_i := V_i \setminus \{x\};\ E_i := E_i \cup \{x\}$

    $\forall \left(x \xrightarrow{a} s'\right) \in succ(x)$

      if $h(s') \neq i$ **then**

        SEND $\left(x \xrightarrow{a} s', h(s')\right)$

      endif

      []

      if $h(s') = i$ **then**

        UPDATE $\left(V_i, E_i, T_i, x \xrightarrow{a} s'\right)$

      endif

  endif

  []

  RECV $\left(s \xrightarrow{a} x\right)$; **UPDATE** $\left(V_i, E_i, T_i, s \xrightarrow{a} x\right)$

endwhile

DMC communication: Time

- Data exchanged:
  - Number of messages (cross arcs, control messages)
  - Data type (handler address, aggregated messages, …)
  - Frequency of exchange (fine or coarse grained computing)
  - Size of messages (user defined, kernel dependent, …)

- Communication cost model: [G. Fox 1989]
  - Monothreaded: $T = T_{compute} + T_{communicate}$,
    $T_{communicate} = N_c(T_s + L_c T_b)$,
  - Multithreaded: $T = max(T_{compute} + N_c T_s,\ N_c L_c T_b)$,

where each of the N_c communications requires time linear in the size of the message ($L_c T_b$), plus a start-up cost (T_s).

DMC communication: Memory

- Huge amount of memory (bottleneck of DMC)
  - to explore and store the state space
- Extensive computation
  - to traverse the graph and to evaluate nodes

➔ Need to reduce the communication overhead to a minimum

- Buffering (network transport, aggregation)
- Multiple communication operations at once (buffering, marshalling, transmitting)
- Asynchronous calls (sending)

Outline

1. Introduction
2. Message Passing mechanisms
3. DMC communication
4. Communication paradigms
5. Conclusion

4. Communication paradigms

Paradigms: Modeling

- 4 criteria (15 possibilities):
  - Synchronous / asynchronous
  - Blocking / non blocking
  - Buffered / unbuffered
  - Bounded buffer / unbounded buffer

- Only 3 models (asynchronous):
  - Blocking communication
  - Non blocking communication with unbounded buffer
  - Non blocking communication with bounded buffer

Paradigms: Blocking communication

- Pros:
  - No buffering, no multiple copy, memory saving
  - More understandable program behavior
  - Short messages directly handled by kernel buffer

- Cons:
  - Complex computation ordering for overlapping
  - Difficult programming for processor cost/performance
  - Synchronization delays (rendez-vous)
  - High deadlock risk

Paradigms: Unbounded buffer

- Pros:
  - Maximal overlapping of communication and computation
  - Maximum flexibility (undelayed transmission calls)
  - Clear program behavior specification
  - Widespread communication mechanisms (MPI, PVM)
  - Majority of DMC papers written with this model
- Cons:
  - Uncontrolled memory resources consumption
  - Uncontrolled buffer overflow (unpredictable behavior, deadlock)
  - Opposite to model checking interest

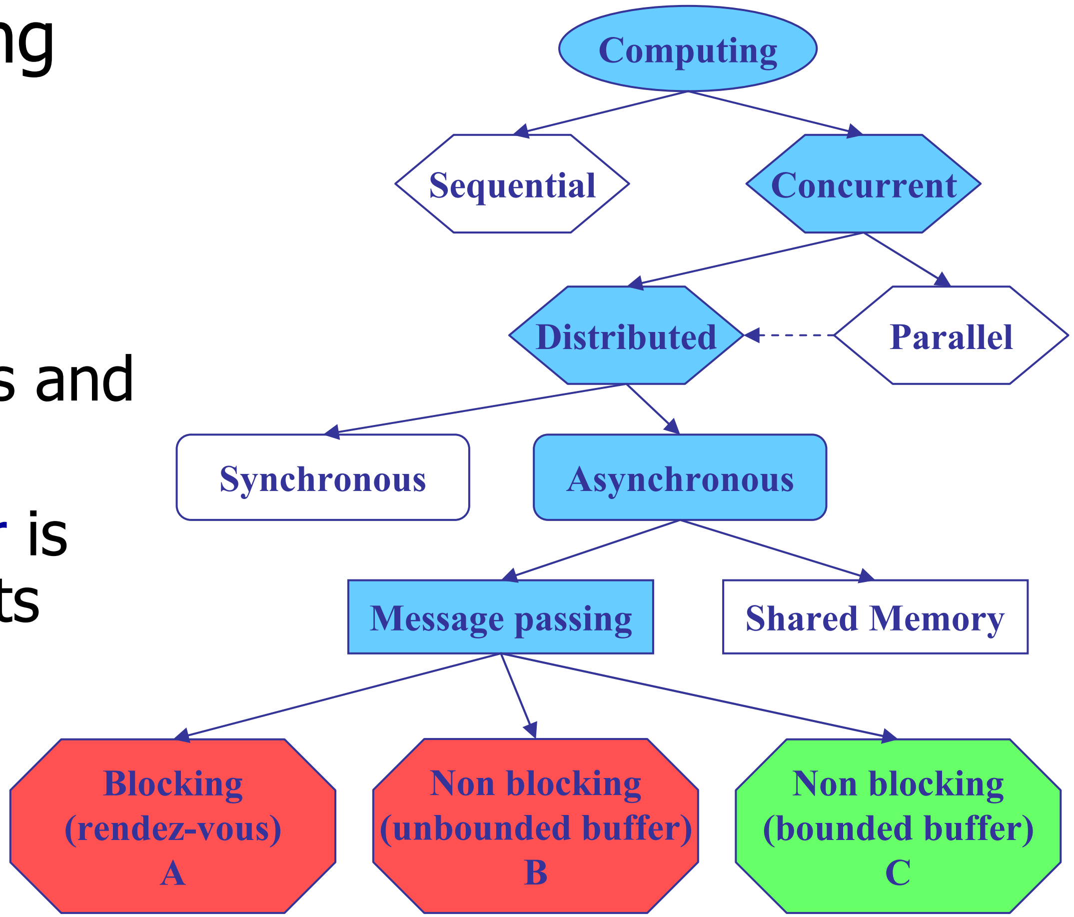
Paradigms: Bounded buffer

- Pros:
  - Interleaving of computations when communication fails
  - Fine use of memory resources
  - Flow control enabled
  - Well-adapted to TCP/UDP sockets over IP
- Cons:
  - Difficult and tricky programming
  - Complex specification
  - Not abstracted in most DMC algorithms

5. Conclusion

Conclusion: Taxonomy

- Distributed computing taxonomy:
  - Advances for each element in DMC tools and algorithms
  - Communication layer is one of these elements
  - Many possible communication paradigms, few practical



Conclusion: Evaluation

- Gap between realistic modelization of process interconnection and concrete implementation
  - Example of the generic distributed state space generation algorithm
- Impact of message passing mechanisms over implementation correctness and performance
- Bounded buffered non blocking communication implemented with TCP/UDP sockets over IP is a good candidate for DMC communication mechanism

Conclusion: Future work

- Basis for DMC communication library implementation
  - Constant evolution and improvements in message passing, but few restrictions always true (installing an extra software, compiling it for each architecture used, learning a new message passing language with too many features for actual works, …)

- Basis for any DMC tools upon precise communication paradigm
  - Subject to experiment different models and to argument paradigm choices
  - Validation of theoretical solution to the problem of DMC communication

Related work

- [A.S. Tanenbaum and M.van Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall, 2002]
  - or any good (undergraduate) book on distributed computing

- [G.Ciardo and D.M. Nicol, *Automated Parallelization of Discrete State-space Generation*, JPDC, 1997]
- [U. Stern and D.L. Dill, *Parallelizing the Murphi Verifier*, CAV'97]
- [B. Haverkort, H. Bohnenkamp and A. Bell, *On the Efficient Sequential and Distributed Evaluation of Very Large Stochastic Petri Nets*, PNPM'99]

- [H. Garavel, R. Mateescu and I. Smarandache, *Parallel state space construction for model-checking*, SPIN'01]

▶More information on:
- http://www.inrialpes.fr/vasy/cadp