DiVinE and DiVinE within

J. Barnat, L. Brim, I. Černá, P. Šimeček, ...

Outline

- Introduction
- Programmer's point of view
- User's point of view
- Future plans

Motivation

Several distributed LTL Model-Checkers

- implemented using various tools
  - `spin`, `maso`, `diks`, ...
- difficult to be used by other users
- incomparable performance
- solve many common problems

DiVinE

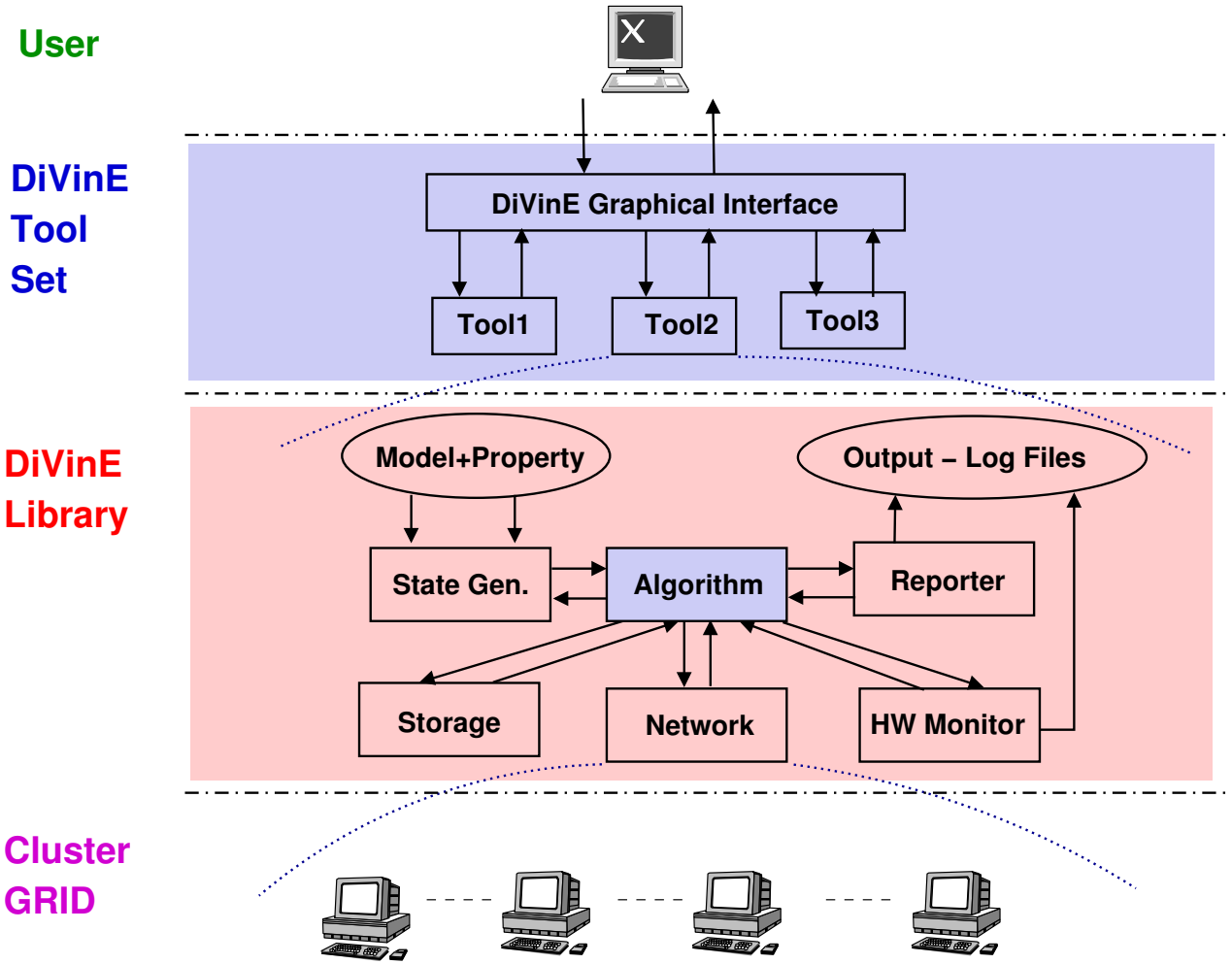
DIVINE

- DIstributed VerIificatioN Environmnet

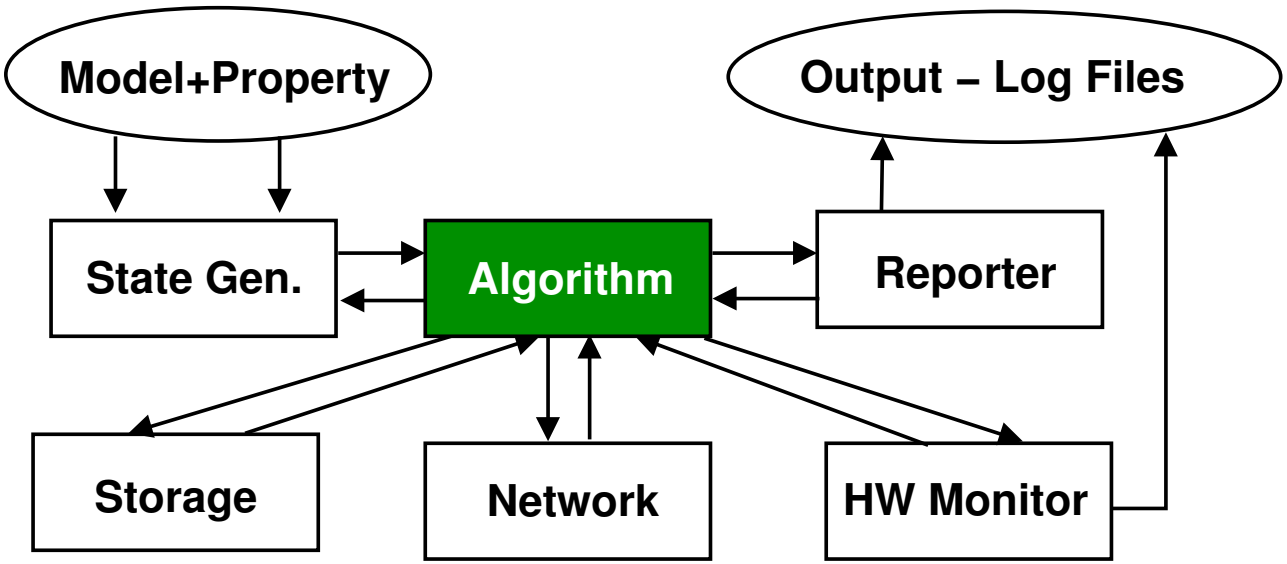
Goals

- Distributed enumerative model-checker
- Development environment
- Platform for experimental evaluation
- Research vehicle

DiVinE Structure



DiVinE from programmer's point of view


Model+Property
Output – Log Files
State Gen.
Algorithm
Reporter
Storage
Network
HW Monitor

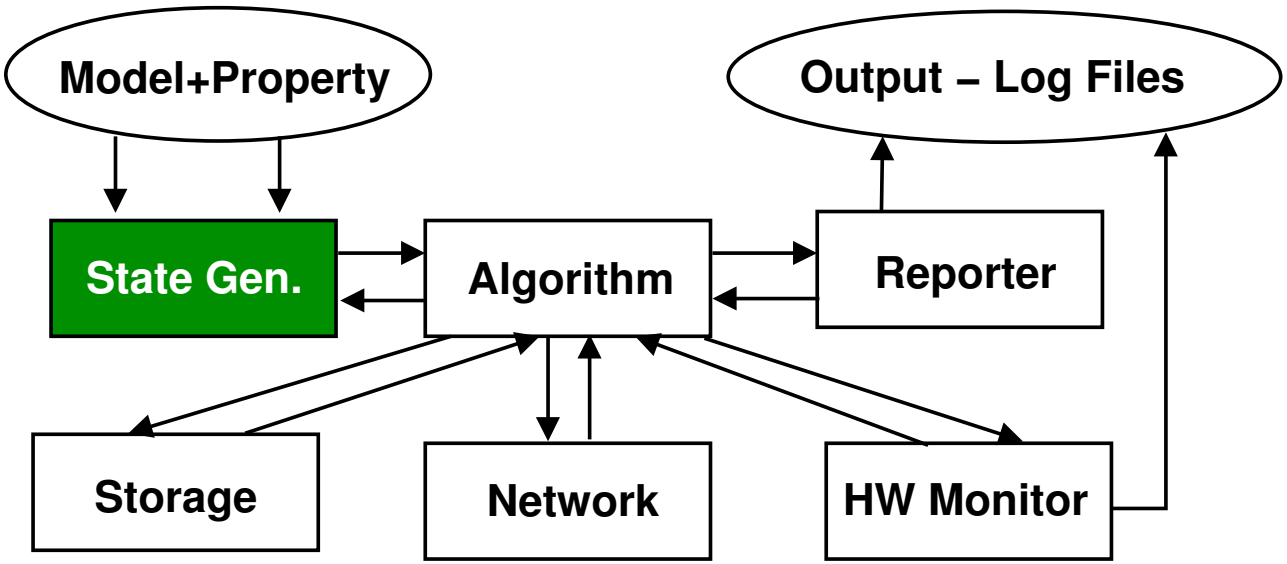
Implementing Algorithm

Box “algorithm”

- compute the model-checking task
- control computation
- call DiVinE Library functions

Other boxes

- arms and legs of the algorithm
- provide more than 100 useful functions


Model+Property
Output – Log Files
State Gen.
Algorithm
Reporter
Storage
Network
HW Monitor

State Generator

Graph of synchronous product automaton

- get_initial_state()
- get_succs()
- is_accepting()

Access to inner structure of the model

- partial order reduction
- property automaton decomposition
- static analysis

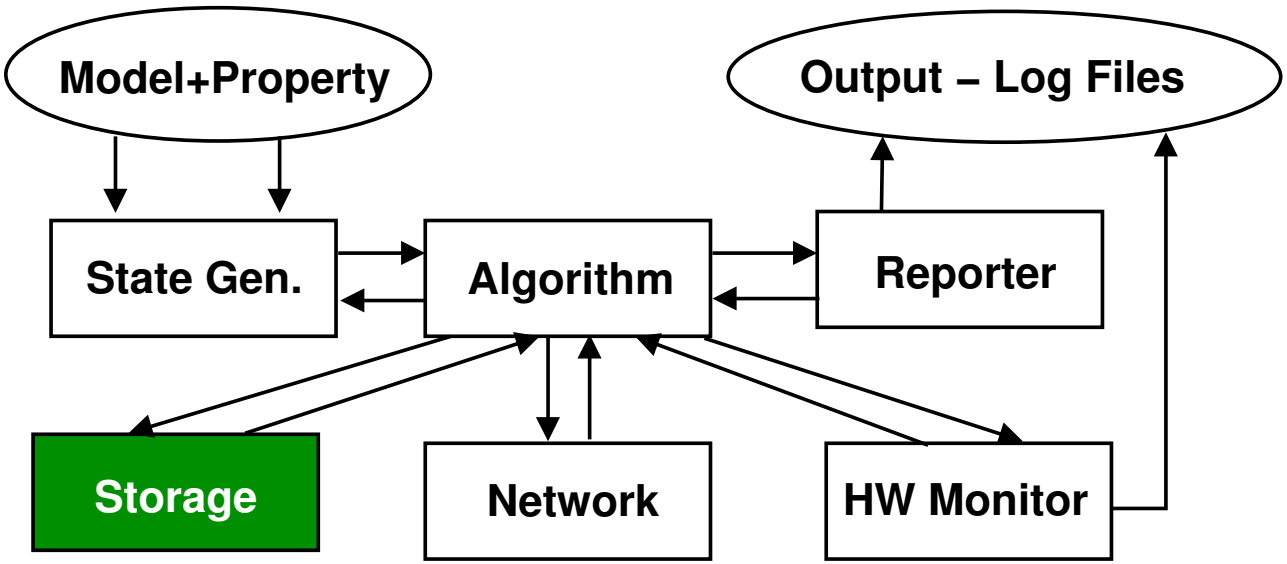
Modeling Languages

DiVinE native modeling language

- another modeling language
- nobody wants to learn

Other modeling languages

- separate system class
- methods to test system abilities
  - can_property_process()
- Promela (NIPS project)


Model+Property
Output – Log Files
State Gen.
Algorithm
Reporter
Storage
Network
HW Monitor

What is it a State

State

- a piece of memory given by generator
- dynamic size

Appendix

- constant-sized piece of information
- associated with every state

Storage

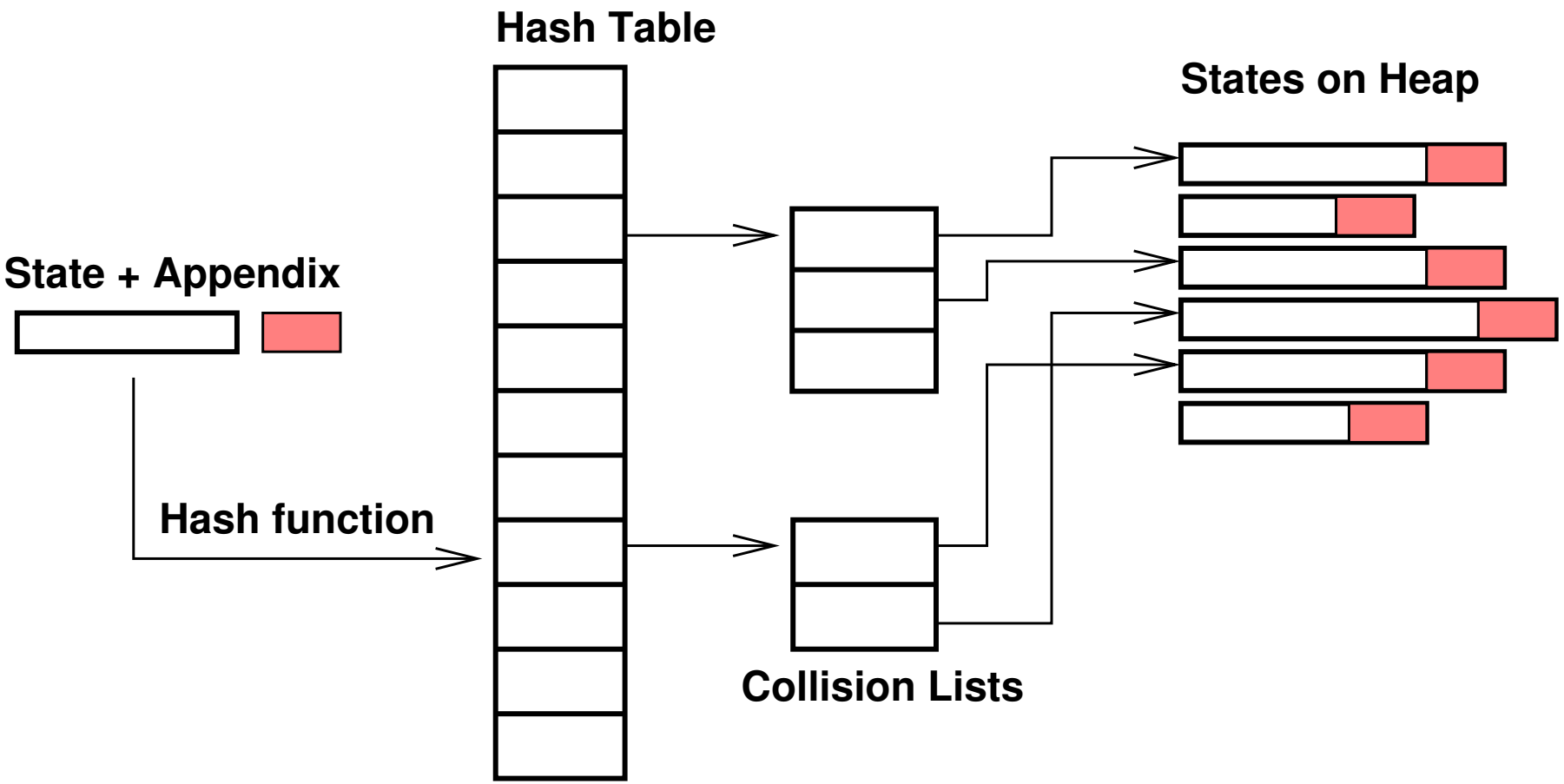
State management

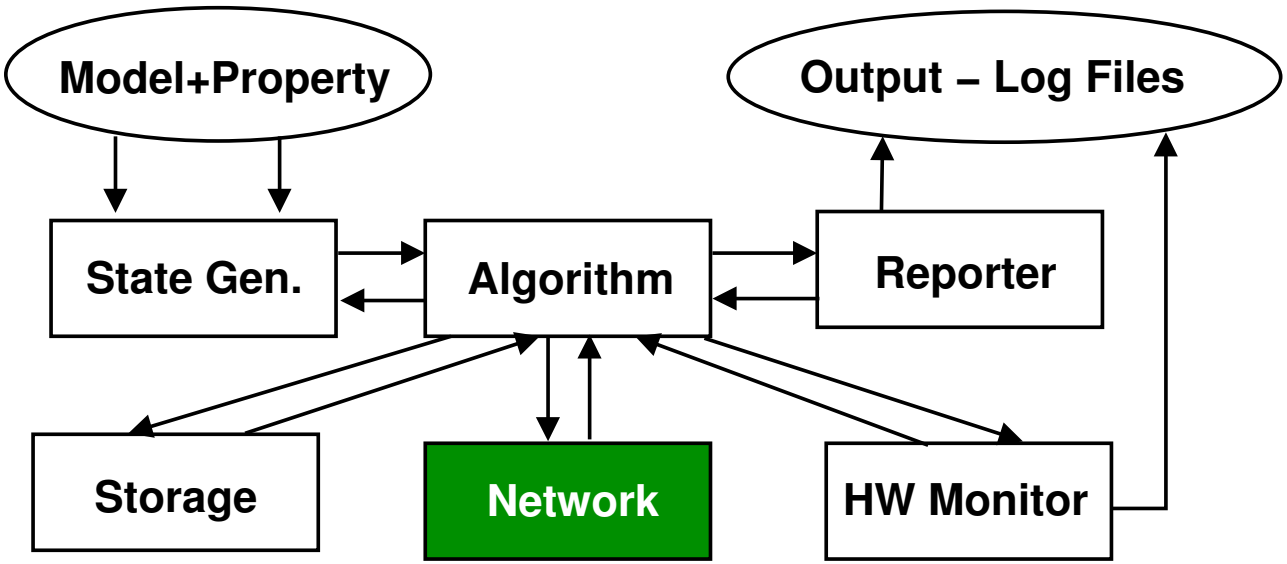
- states organized using hash table
- standard state manipulation functions
- 8 byte state reference

State compression

- no compression
- static Huffman's encoding

Storage structure




Model+Property
Output – Log Files
State Gen.
Algorithm
Reporter
Storage
Network
HW Monitor

Network Support

Basic network primitives

- send (urgent) message
- barrier synchronization

Receives

- procedure to process user messages
- `process_messages()`

Network Support – cont.

Distributed termination detection

- Safra's algorithm
- busy/idle state
- performed within `process_messages()`
- test for being synchronized
- can exchange data within synchronization
- repeatable

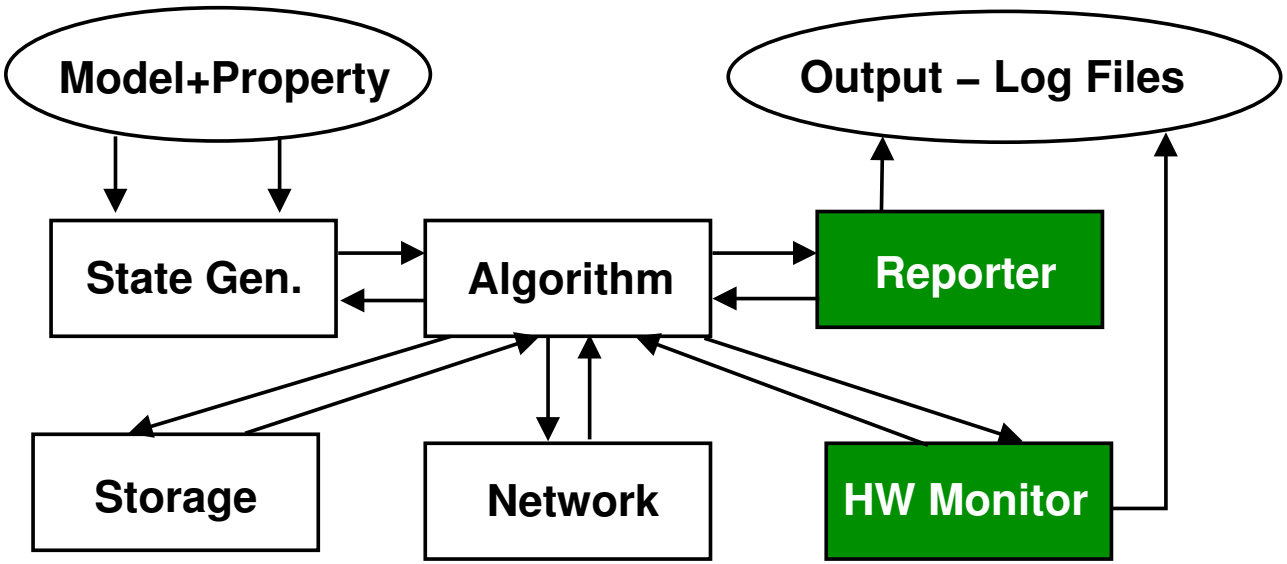
Network Support – cont.

Others

- additional buffers
- partition function
- network statistics
- wrapper for direct network access

Network Support – Example

```
void process_message(char *buf, int size, int src, ...) {
  state_t state = new_state(buf,size);
  if (!Storage.is_stored(state) {
    Storage.insert(state,state_ref);
    Queue.push(state_ref);
    Distributed.set_busy(); } }

Distributed.process_user_message = process_message;

state_t state = System.get_initial_state();
if ( Distributed.partition_function(state) == my_id) {
  Storage.insert(state,state_ref);
  Queue.push(state_ref); }

while (!Distributed.synchronized()) {
  Distributed.process_messages();
  while (!Queue.empty()) {
    state_ref = Queue.top(); Queue.pop();
    state = Storage.reconstruct(state_ref);
    System.get_succs(state, succs);
    for (size_int_t i=0; i!=succs.size(); ++i) {
      int owner = Distributed.partition_function(succs[i]);
      Distributed.send_message(succs[i].ptr,succs[i].size,...)
      } }
  Distributed.set_idle(); }
```


Model+Property
Output – Log Files
State Gen.
Algorithm
Reporter
Storage
Network
HW Monitor

Output Classification

Type of output

- final result and statistics
- runtime statistics
- error messages

Algorithm dependency

- dependent
  - queue sizes, number of iterations, ...
- independent
  - number of stored states, sent messages, ...

Output Classification – cont.

Idea

- `stdout` belongs to algorithm
- other outputs written to files

Advantages and disadvantages

- persistent
- unified format
- further processing
- slowdown

Output – Solution

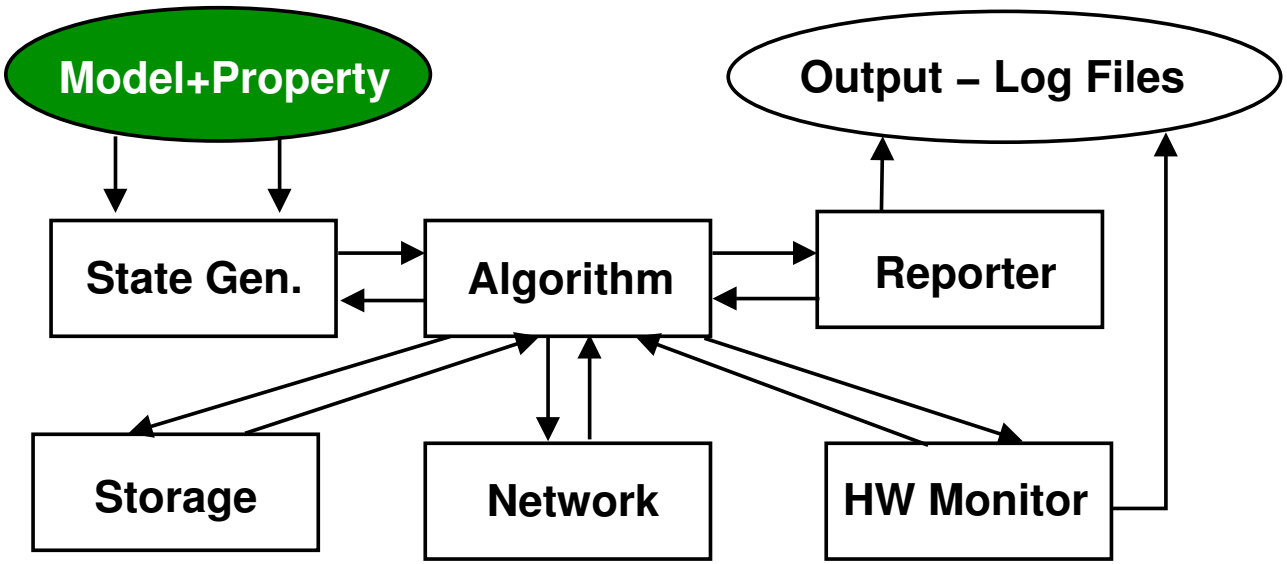
Final report

- single file
- produced before network is finalized

Logfiles

- one file per each workstation
- produced during computation
- POSIX signal + UNIX alarm utility

DiVinE from user's point of view


Model+Property
Output – Log Files
State Gen.
Algorithm
Reporter
Storage
Network
HW Monitor

DiVinE Native Modeling Language

Processes

- extended FA
- transitions with *guards*, *sync*, *effects*

Interprocess communication

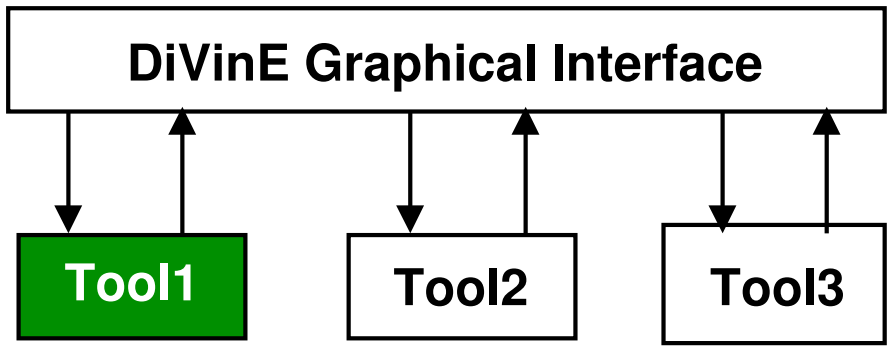
- shared variables
- buffered/unbuffered channels

System

- synchronous, asynchronous
- property Büchi automaton

DiVinE Native Modeling Language – Example

```
...
process cabin
{
  state idle,mov,open;
  init idle;
  trans
  idle -> mov  {guard v>0;},
  mov -> open  {guard t==p;},
  mov -> mov   {guard t<p; effect p=p-1;},
  mov -> mov   {guard t>p; effect p=p+1;},
  open -> idle {effect req[p]=0,v=0;};
}
...
system async property LTL_negative_claim;
```


DiVinE Graphical Interface
Tool1
Tool2
Tool3

DiVinE ToolSet

How to make it work

- download from our website
- compile
- run (`divine.*`)

Prerequisites

- Linux cluster
- MPI

Using DiVinE Tool

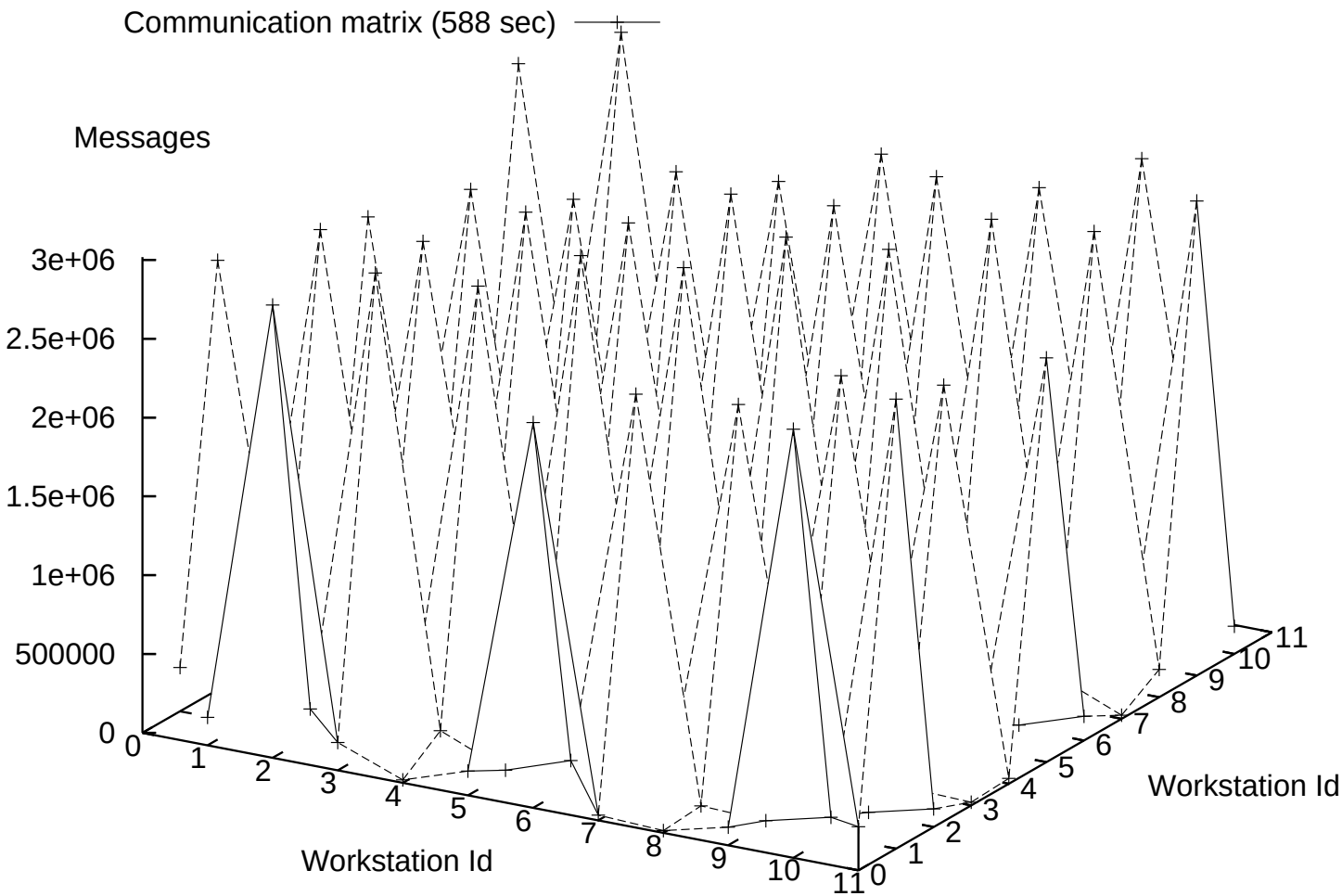
divine.owcty

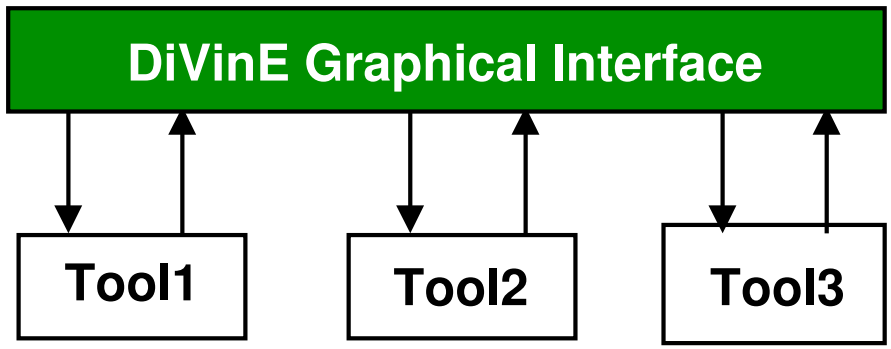
```
DiVinE Tool Set

OWCTY version 1.0 build 4  (2005/09/21 17:14)

Usage: [mpirun -np N] divine.owcty [options] input_file
Options:
 -V,--version              show version
 -h,--help                 show this help
 -H x,--htsize x           set the size of hash table to
                           ( x<33 ? 2^x : x )
 -v,--verbose              print some statistics
 -q,--quiet                quite mode
 -t,--trail                produce trail file
 -r,--report               produce report file
 -s,--simple               perform simple reachability only
 -L,--log                  produce logfiles (log period 1 sec)
 -X w                      sets base name of produced files to w
                           (w.trail ,w.report ,w.00-w.N)
```

DiVinE ToolSet – More than Algorithms

- Reachability
  - deadlocks, goal states, unreachable code
- Simulator
- `divine.ltl2buchi`
- Utility to draw state-space
- Utility to visualize logfiles
- Initial set of parametrized models

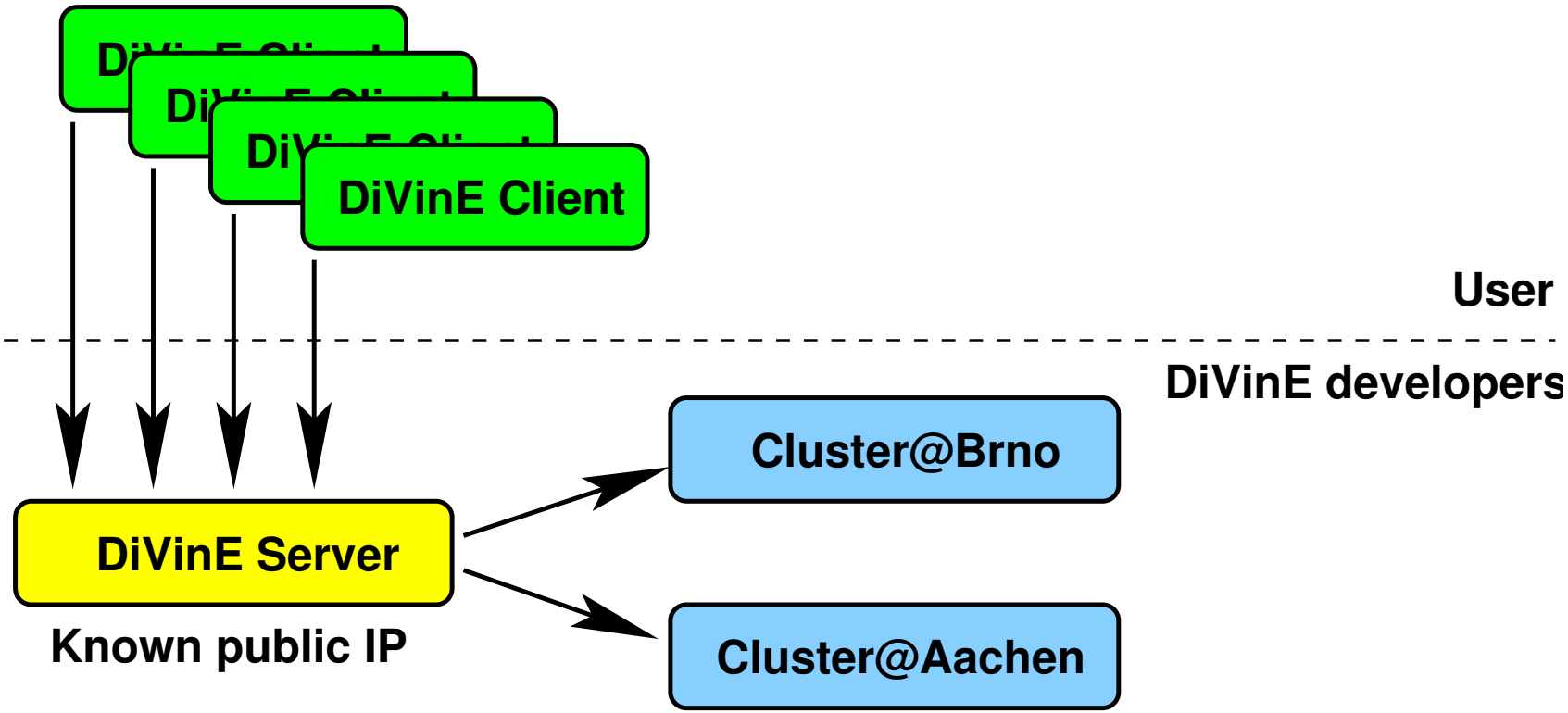
DiVinE ToolSet – Visualized Output Files



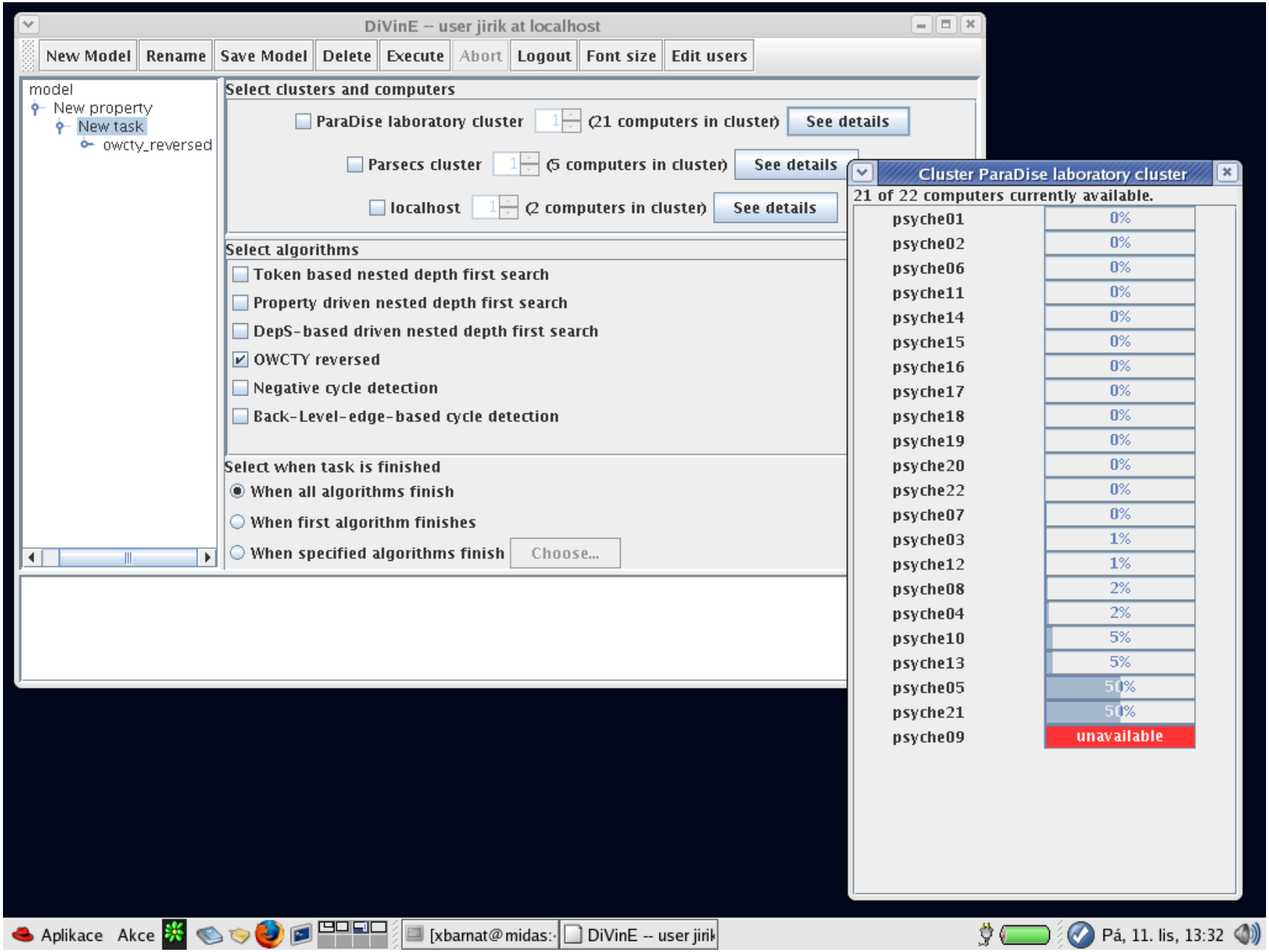

DiVinE Graphical Interface
Tool1
Tool2
Tool3

Graphical User Interface (GUI)

Server-client application



A very short demo



Future plans

General goals

DiVinE as a tool

- stabilize GUI
- extend functionality

DiVinE for programmers

- improve design of library
- optimize implementation
- documentation
- develop and implement new ideas

Load Balancing

Dynamic load-balancing

- memory occupation
- work load
- network load

Known techniques

- states partitioning/repartitioning
- queue balancing

Queue balancing

Idea

- states to be explored on highly-loaded workstations explored on less-loaded workstations

Problem

- how to access appendix on remote workstations?

URL

http://anna.fi.muni.cz/divine


DiViNE