VGrADS
Virtual Grid Application Development Software Project

GridSAT: Solving Problems of Propositional Satisfiability using Cyberinfrastructure

Rich Wolski

Wahid Chrabakh

University of California, Santa Barbara

Powered by

The VGrADS Project

SDSC

The NPACI Grid

GridSAT: Early Attempt at Demonstrating CI

VGrADS
Virtual Grid Application Development Software Project

- **Application: Automatic Solver for Propositional Satisfiability problems**
  - Given a boolean formula, is there an assignment of values to variables such that the formula is satisfied?
    - Circuit design and verification, cryptography, logistics and scheduling, reliability and model verification
- **From Research to Infrastructure and back Again**
  - **Start** from first-principles and design Grid-enabled solver (research)
  - **Build** using state-of-the-art **VGrADS** programming methods (research)
  - **Test** using "local" project resources (research)
  - **Transition** successful technology to production middleware (infrastructure)
  - **Deploy** to couple production and project resources and run (infrastructure)
  - **Rise and repeat** as necessary to produce new science

Classic Satisfiability

VGrADS
Virtual Grid Application Development Software Project

- **Set of variables $V=\{v_i \mid i=1,\ldots,k\}$**
- **Literal: a variable or its complement**
- **Clause: OR of a set of literals**
  - For example: $C_j = (v_a \,||\, v_b \,||\, v_c \ldots)$
- **Conjunctive Normal Form equation**
  - $X=C_1 C_2 C_3 \ldots C_k$
- **SAT Problem:**
  - Does there exist an assignment of boolean values to variables that makes the equation true?
- **NP-Complete**

SAT Competition

VGrADS
Virtual Grid Application Development Software Project

- **Held annually to test the power of automatic SAT solvers**
  - Community benchmarks: : www.satlive.org/SATCompetition/
  - 25 categories, 832 problems total
  - Solvers ranked based on overall score (relative solver performance varies depending on problem instance).
  - **zChaff** is the current world record holder
    - High performance, sequential implementation of Chaff algorithm

zChaff

VGrADS
Virtual Grid Application Development Software Project

- **Chaff Algorithm General characteristics:**
  - New *clauses* are learned as solver progresses.
  - Clauses stored in memory in large database.
  - Data and compute intensive.
  - Expendable clauses are discarded to keep database from overgrowing available memory.
- **zChaff optimizations:**
  - Public domain from S. Malik's group at Princeton
  - Detects clauses important to the current search step faster
  - Better algorithm for evaluating which variable assignment would have the most impact on search process
- **Thought to be impossible to distribute**
  - Fundamentally a tightly coupled database code
  - Clause sharing of unknown value

GridSAT

VGrADS
Virtual Grid Application Development Software Project

- **Distributed, Parallel zChaff: Many processes exploring different disjoint parts of the search space**
  - New clause sharing algorithm -- not all clauses are needed by each process
  - Pertinent newly learned clauses are shared for solver efficiency
- **Grid-enabled zChaff: sharing and distribution algorithm "tunes" itself based on dynamically fluctuating resource availability**
  - Available core memory
  - Network connectivity
  - CPU load
- **Core clause database expands and contracts as resource availability and fluctuates**
  - Not your father's MPI code

GridSAT Resource Scheduling

VGrADS
Virtual Grid Application Development Software Project

- **Scheduling model: parallelism should be avoided**
  - Tightly coupled database code => database must be fragmented and replicated on the fly
- **Scheduling algorithm**
  - Start with one node
  - When node becomes "exhausted", identify one additional node to share the load
  - Split the workload between exhausted node and fresh node (grow the node set by one)
  - If a node backtracks through an irresolvable conflict, free the node (shrinking the node set by one)
- **Apply the scheduling algorithm recursively to each node**
  - The application "**grows**" into the resources and aggressively **shrinks** back in order to minimize communication overhead

When and Where to Schedule

VGrADS
Virtual Grid Application Development Software Project

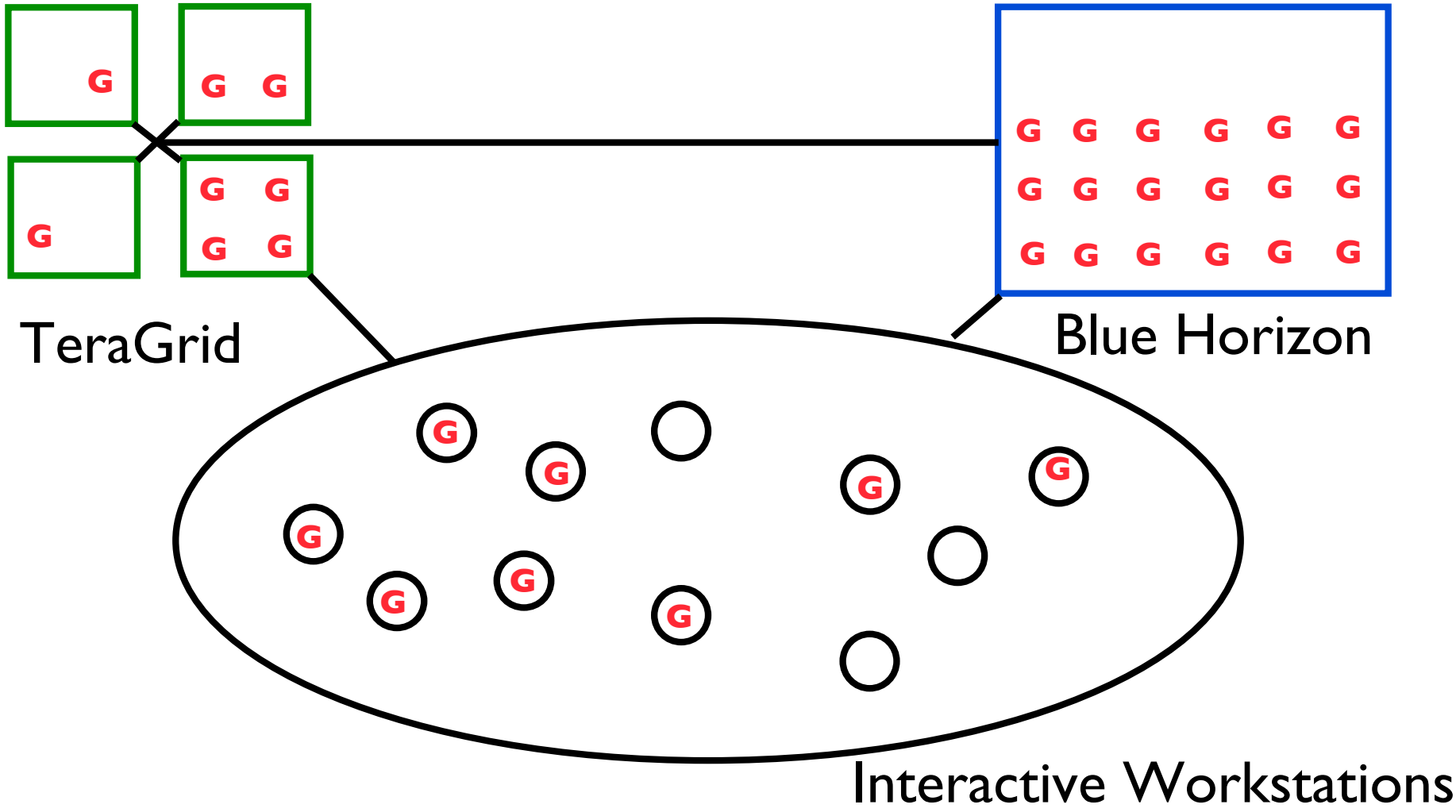
- **When: a node becomes exhausted when**
  - The internal database grows beyond the available memory
  - The problem has executed longer than the time needed to replication the relevant portions of its database to a "best" candidate additional node
- **Where: choose the node with the best weighted sum of**
  - Predicted available memory
  - Predicted CPU availability
  - Predicted network connectivity to the splitting node
  - Predicted network connectivity to "good" neighbors
- **Dynamic resource predictions are generated by the Network Weather Service**
  - NMI and NPACKage distribution
  - Core NPACI Grid service

Active Queuing

VGrADS
Virtual Grid Application Development Software Project

- **GridSAT scheduler understands batch queue submission and space-sharing**
  - Launches and periodically relaunches batch requests on batch-controlled resources
  - When the resources come on-line, they report to the the scheduler
  - The scheduling algorithm **migrates** work from slower machines to the faster space-shared processors
  - When the time limit expires, the work is migrated back out
  - Checkpoints store work backlog until enough processors become available again
- **Jobs running on slower resources make progress while batch jobs are in queue => active queuing**
- **No co-scheduling or advanced reservations are necessary**

For Example



First Step: GrADSAT

VGrADS
Virtual Grid Application Development Software Project

- **Goal: test VGrADS "ease of programming" capabilities**
  - *Can we build abstractions that make a new class of applications possible?*
- **Research prototype**
  - Use GrADSoft to build an experimental implementation
  - Test using **VGrADS** resources
    - Project-wide clusters and desktop workstations
    - Non-dedicated access: must be shared by all participants (including people sitting at the desks where the desktops are located)
- **Infrastructure enhancements**
  - GrADSAT exposed many opportunities for improving and optimizing NMI and NPACKage infrastructure components
- **Algorithmic improvements**
  - Clause sharing verified to be a big win (new CS result)

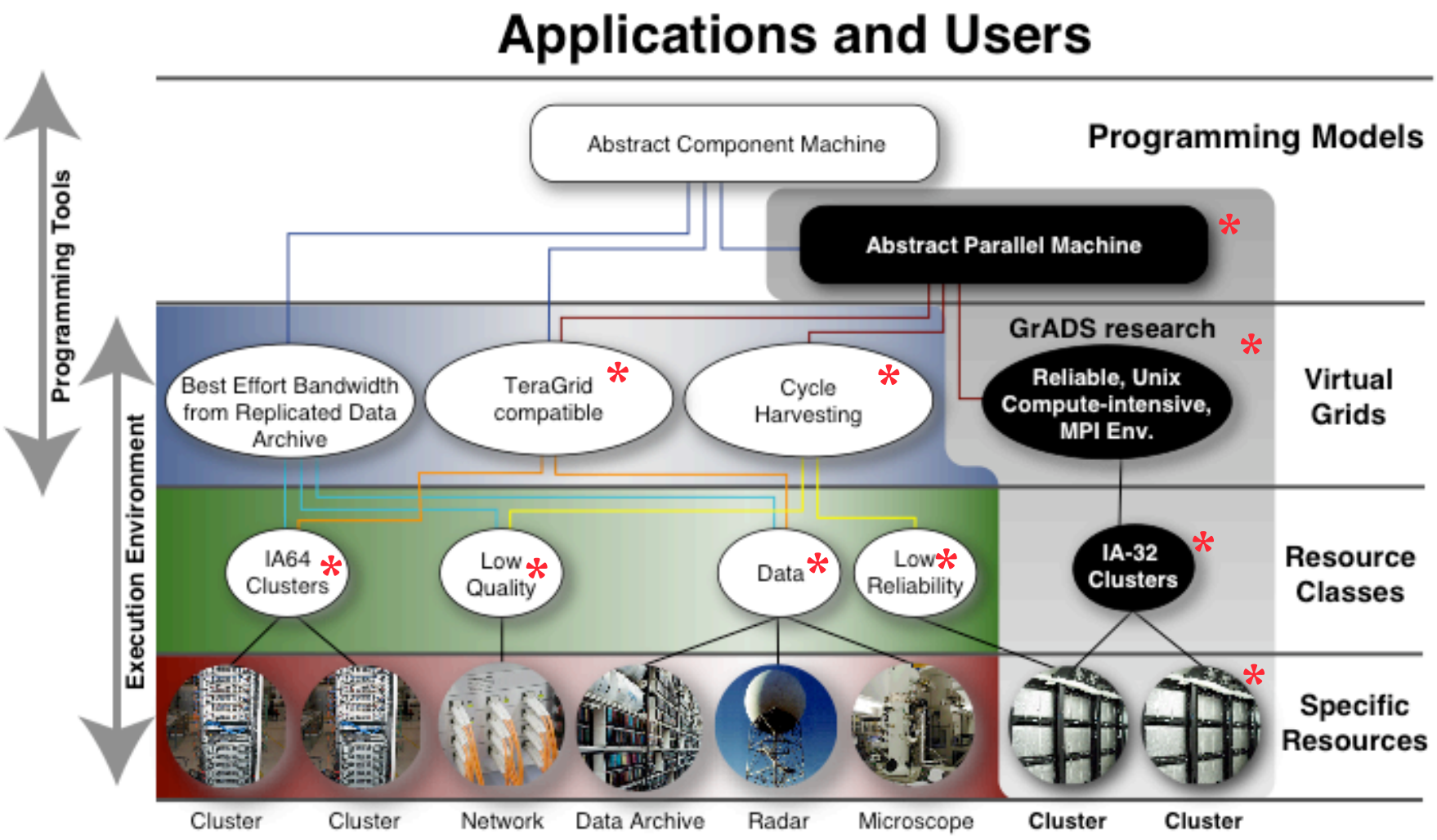
Next Step: GridSAT

VGrADS
Virtual Grid Application Development Software Project

- **Combine project resources (VGrADS testbed) with production HPC resources (Datastar, TeraGrid, etc.)**
- **Uses EveryWare**
  - Minimalist programming toolkit for building applications as distributed services atop different infrastructures
  - Compatible with Globus, NPACI Grid, GrADSoft, Condor, IBP, etc.
  - Based on Network Weather Service internal infrastructure
- **Added checkpointing**
  - Uses the **Logistical Backbone** and the **Internet Backplane Protocol** (Plank, Beck, Dongarra) as a checkpoint appliance
  - Local disk footprint minimal
  - Approximately 250 repositories => Planetary operation
- **Goal: New domain science**
  - Early GrADSAT results indicated substantial performance improvement

"Seeing" the Vision

VGrADS
Virtual Grid Application Development Software Project



GridSAT "Live"

VGrADS
Virtual Grid Application Development Software Project

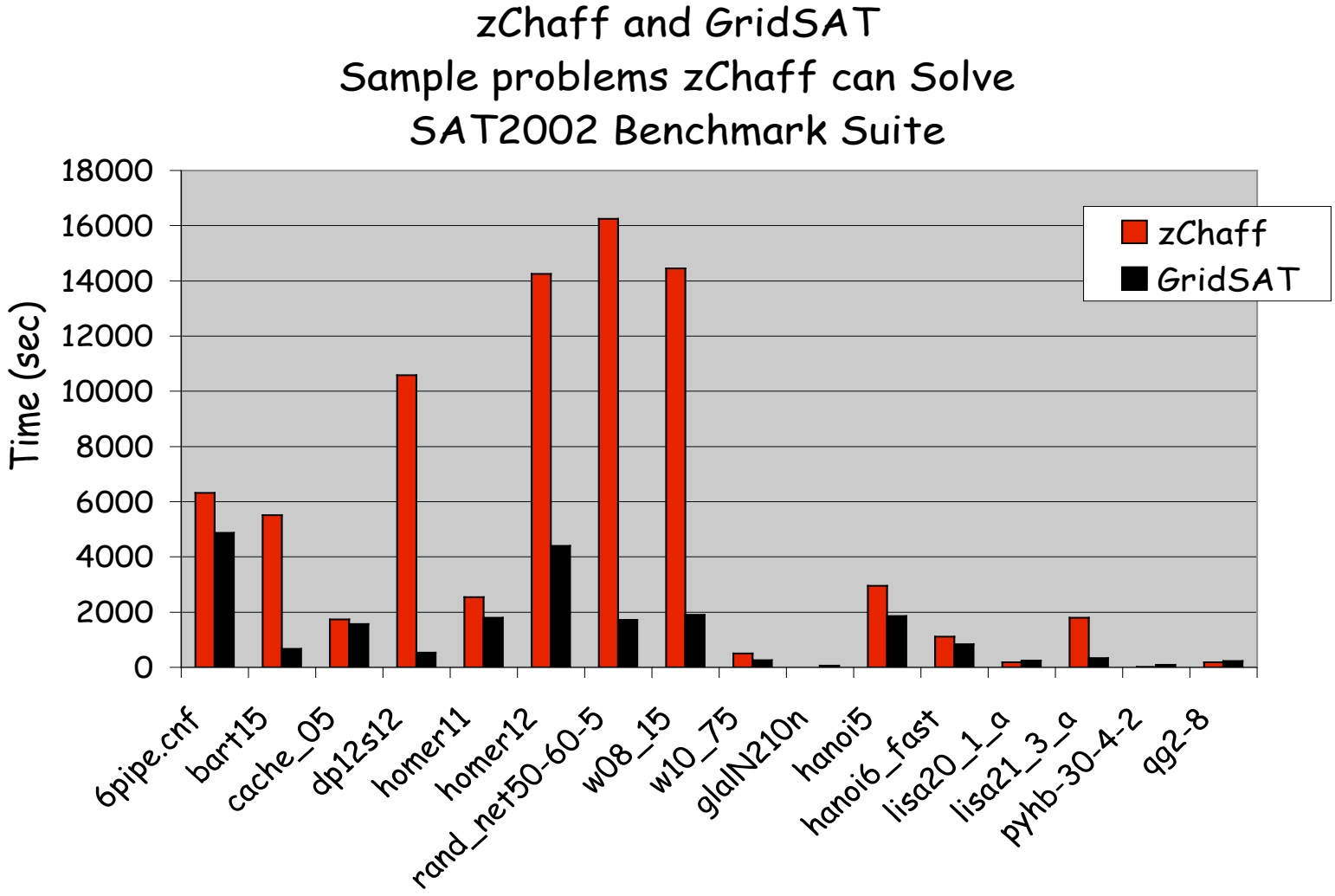
- GrADS Testbed
- SDSC Datastar
- TeraGrid (SDSC, NCSA, Caltech, Argonne)
- Cray SV1
- UCSB MAYHEM Lab
- HPC Challenge Testbed

GridSAT vs. zChaff

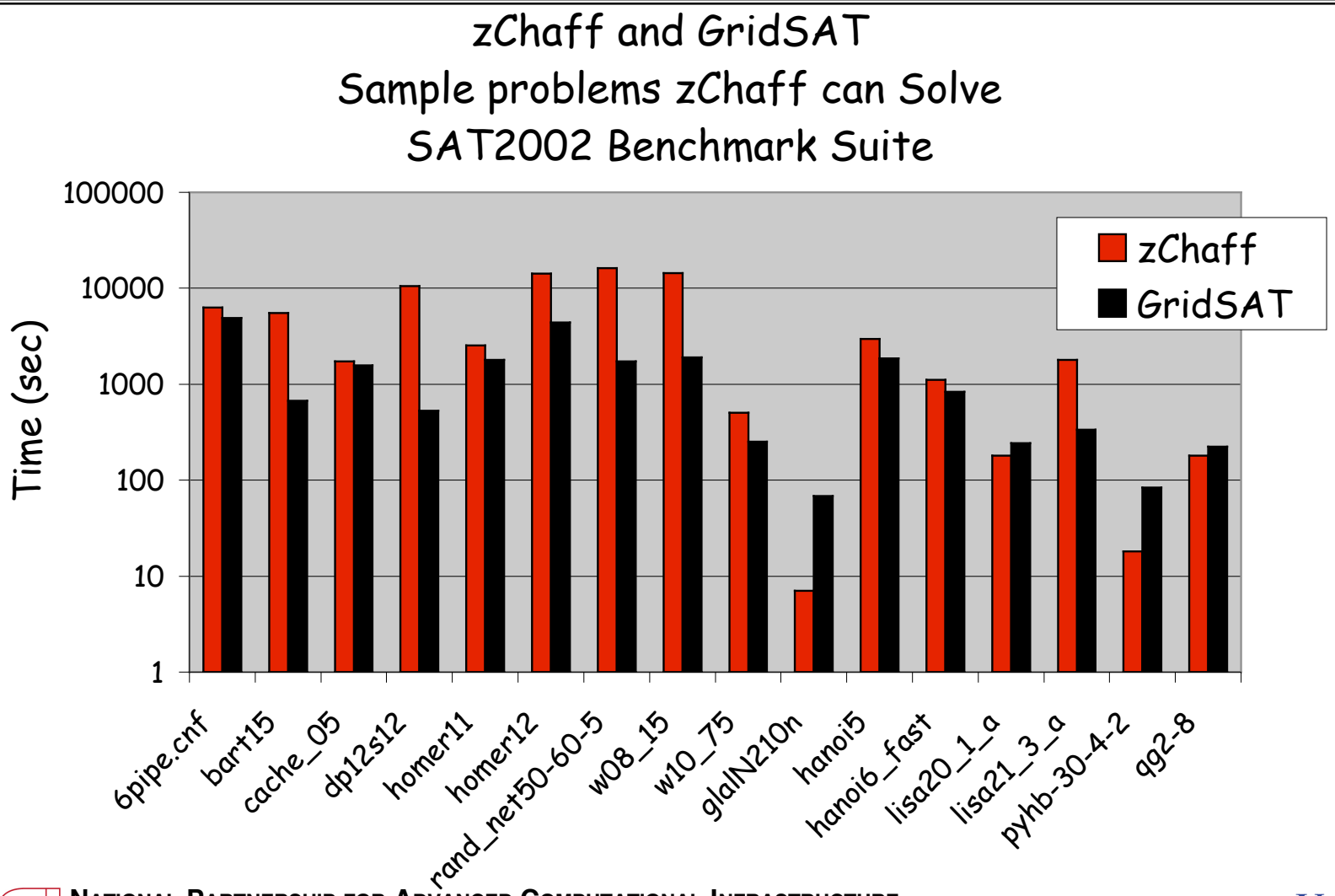
VGrADS
Virtual Grid Application Development Software Project

- **Test using problems zChaff is able to solve**
- **GridSAT using GrADS testbed in non-dedicated mode**
  - EveryWare over GrADSoft
  - Four clusters, misc. workstations, approximately 64 processors
  - CPU speed: 233 MHz to 900 MHz
  - CPU memory: 128M to 1GB
  - UTK, UIUC, Rice, UCSD, UCSB
- **zChaff using fastest single machine in dedicated mode**
  - 900MHz P4 with 1GB memory

Performance Comparison



Log Scale


zChaff and GridSAT
Sample problems zChaff can Solve
SAT2002 Benchmark Suite
Time (sec)
100000
10000
1000
100
10
1
zChaff
GridSAT
6pipe.cnf
bart15
cache_05
dp12s12
homer11
homer12
rand_net50-60-5
w08_15
w10_75
glaIN210n
hanoi5
hanoi6_fast
lisa20_1_a
lisa21_3_a
pyhb-30-4-2
qg2-8

GridSAT offers substantial speed improvement

VGrADS
Virtual Grid Application Development Software Project

- **Parallelism and clause sharing a big win**
  - Community is "surprised" => German SAT company
- **Scheduling chooses the "right" resources**
- **GridSAT slower on problems that complete in 120 seconds or less**
  - Extra machinery implies execution overhead
  - Performance improvement scales with problem "difficulty"

Unsolved Community Problems

VGrADS
Virtual Grid Application Development Software Project

- **GridSAT applied to "challenge" problems unsolved by other solvers (including zChaff)**
- **Submitted to SAT competition**
  - FPGA: routing/layout
  - MC: Model Checking
  - IF: Integer factorization
  - PF: Primes factorization
  - G: Glassy model from statistical physics
  - DP: Dining philosophers
  - RC: Random circuits
  - QG: Quasi group
  - MAT: Matrix multiplication
  - C: Computability
  - RG: randomly generated
  - PAR: parity function
- **Couple Production Resources to GrADS testbed**

New Domain Science

VGrADS
Virtual Grid Application Development Software Project

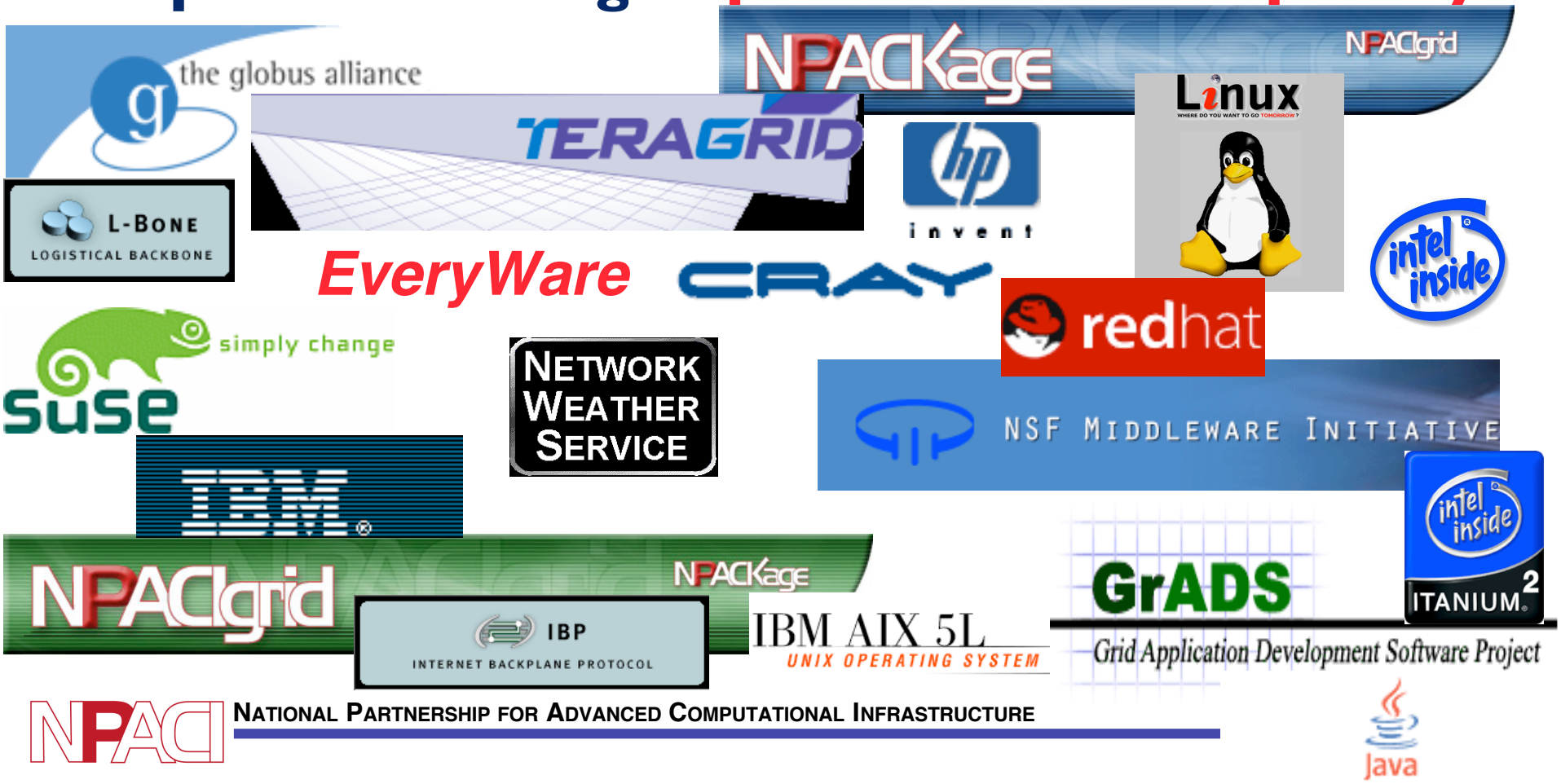
Benchmark name	Category	Solution	zChaff (sec)	GridSAT (sec)
7pipe_bug	FPGA	SAT	x	5058
dp10u09	DP	UNSAT	x	2566
rnd40-60-10	RC	UNSAT	x	1690
f2clk_40	**MC**	**UNSAT***	**x**	**3304**
Mat26	MAT	UNSAT	x	1886
7pipe	FPGA	UNSAT	x	6673
comb2	**MC**	**UNSAT***	**x**	**9951**
phyb-40-4-1	PF	UNSAT	x	2425
pyhb-40-4-2	PF	UNSAT	x	2564
rand_net70-25-5	RC	UNSAT	x	30837
glassybp-v399-s499089820	G	SAT	x	5472
cnt10	**C**	**SAT**	**x**	**4h**
par32-1-c	**PAR**	**SAT**	**x**	**41h**
k2fix-gr-rcs-w8	**FPGA**	**UNSAT***	**x**	**23h**
k2fix-gr-rcs-w9	**FPGA**	**UNSAT***	**x**	**14.3 days**
w08_15	**MC**	**SAT***	**x**	**3141**

***problem solution was previously unknown**

Order out of Chaos

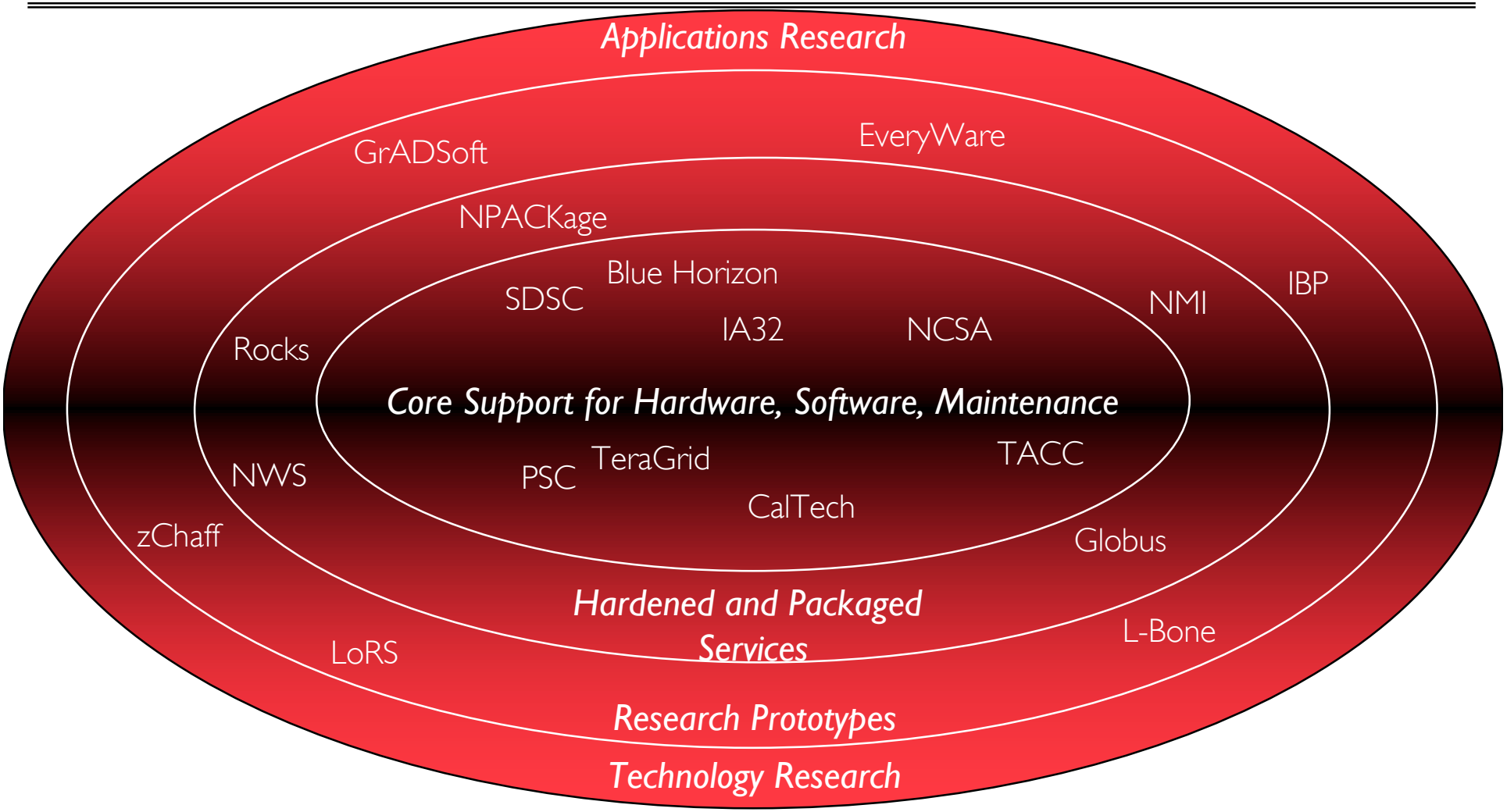
VGrADS
Virtual Grid Application Development Software Project

- **The Grid can enable new science, but new capabilities bring unprecedented complexity**



Cyberinfrastructure

VGrADS
Virtual Grid Application Development Software Project



Conclusions

VGrADS
Virtual Grid Application Development Software Project

- **Grid computing can do more than provide support for legacy applications**
  - **GrADS**: new programming methods that abstract the dynamic and heterogeneous characteristics of Grid systems
  - EveryWare: programs structured as application-specific services using robust underlying infrastructures
- **New algorithms that take into account dynamic resource acquisition and release lead to new science**
- **The number of different hardware, software, and administrative resources that must be harnessed is *staggering***
- **Building cyberinfrastructure support "out" from a core of tested technologies structures the chaos**
- **Effective applications and technology R&D "circulates" from the outside to the core and back**

What's Next?

VGrADS
Virtual Grid Application Development Software Project

- **More science**
  - GridSAT can use a planetary collection of resources
  - L-Bone for checkpointing
  - Supercomputers, Condor, Grid testbeds, Clusters, desktops
  - Crack unsolved SAT problems from the industrial and research communities
- **More Research**
  - Driving application for EveryWare
  - ***VGrADS*** -- virtualization of **GrADS** approach
- **More Users**
  - SAT Gateway coming soon

Thanks[10]

VGrADS
Virtual Grid Application Development Software Project

- **Grid Application Development Software Project**
  - Francine Berman, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, Lennart Johnsson, Ken Kennedy, Carl Kesselman, Dan Reed, Linda Torczon, Rich Wolski
  - Holly Dail, Brett Ellis, Celso Mendes
- **NPACI and SDSC**
  - Nancy Wilkins-Diehr
- **LoCI Lab at UTK**
  - James Plank, Micah Beck, Scott Atchley
- **MAYHEM Lab at UCSB**
  - Graziano Obertelli, Todd Bryan, Larry Miller
- **The TeraGrid Project**