Enabling Massive Parallelism for Stochastic Optimization Problems

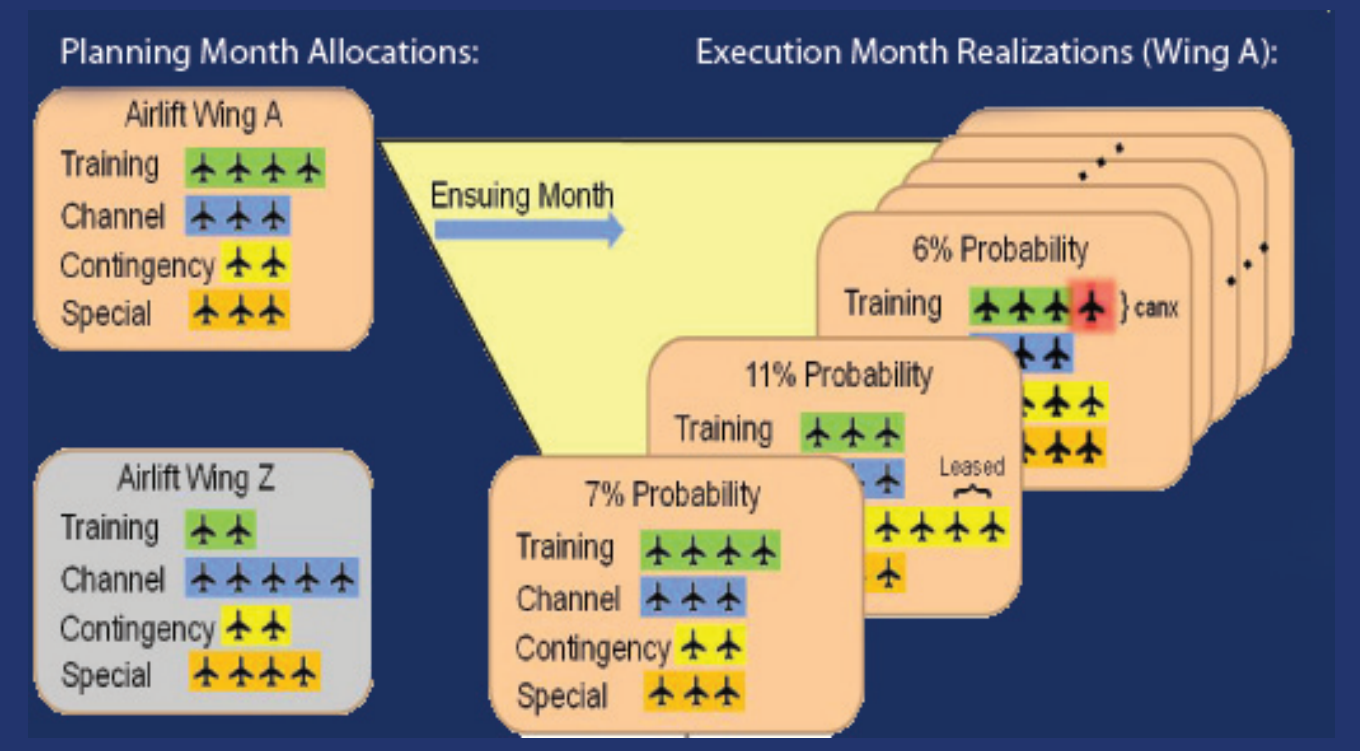
Akhil Langer, Ramprasad Venkataraman, Gagan Gupta, Laxmikant Kale, Udatta Palekar
University of Illinois at Urbana-Champaign
{alanger, ramv, gagan, kale, palekar}@illinois.edu

Steve Baker, Mark Surina
MITRE Corporation
{sbaker, msurina}@mitre.org

Problem Context

Task: Allocate aircraft to cargo and crew delivery missions
Target: Minimize operating costs in the face of uncertain demands

The US air fleet is responsible for moving cargo all over the world, often in the face of sudden events like natural disasters, conflict, etc. The penalty for late or missed deliveries are often steep. Fleet management consists of periodically allocating aircraft and personnel to different cargo delivery missions while allowing for uncertain and sudden demands. Sudden reallocation of craft from one mission to another is also very expensive. The objective is to minimize operating costs by a weighted consideration of a variety of possible scenarios when making an allocation decision.



Two Stage Stochastic Program

Modeling Approach

Stage 1: Compute potential allocation
Stage 2: Evaluate costs of allocation for different scenarios
Iteratively use feedback from all scenarios to refine allocation

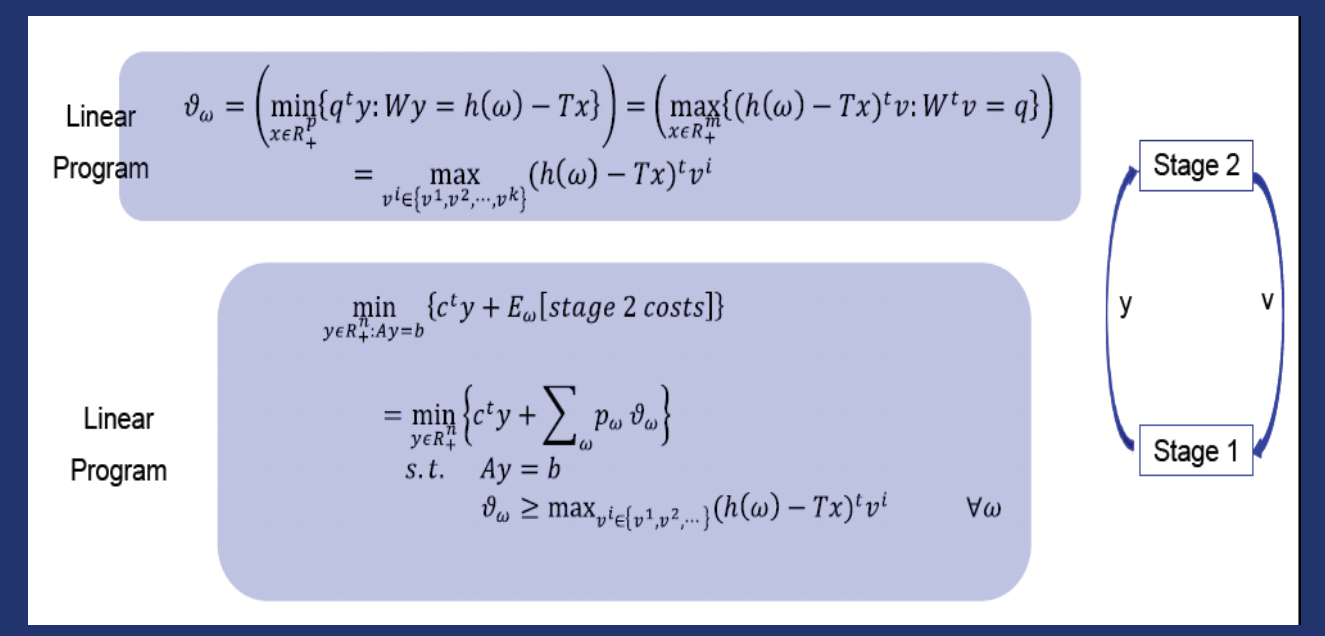
Objective Functions

Minimize:

1. The costs of allocating owned and long term leased aircraft to mission categories (stage 1)

+

2. The expected costs of short-term aircraft leasing, aircraft operating and late and non-delivered cargo and missed missions (stage 2)



$$\vartheta_\omega = \left(\min_{x\in R_+^n}\{q^t y : Wy = h(\omega) - Tx\}\right) = \left(\max_{x\in R_+^m}\{(h(\omega)-Tx)^t v : W^t v = q\}\right) = \max_{v^i\in\{v^1,v^2,\dots,v^k\}} (h(\omega)-Tx)^t v^i$$

$$\min_{y\in R_+^n : Ay=b} \{c^t y + E_\omega[\text{stage 2 costs}]\}$$

$$= \min_{y\in R_+^n}\left\{c^t y + \sum_\omega p_\omega \vartheta_\omega\right\}$$
$$\text{s.t.}\quad Ay = b$$
$$\vartheta_\omega \geq \max_{v^i\in\{v^1,v^2,\dots\}} (h(\omega)-Tx)^t v^i \qquad \forall\omega$$

Coarse Grained Computations

- Linear Programs cannot be broken down trivially
- LPs are delegated to numeric library
- Form fundamental grain of computation

Why Parallel?

- Need to evaluate multiple independant scenarios
- Desired time to solution is typically fixed
- Parallelism enables consideration of more scenarios and greater confidence in resultin allocations

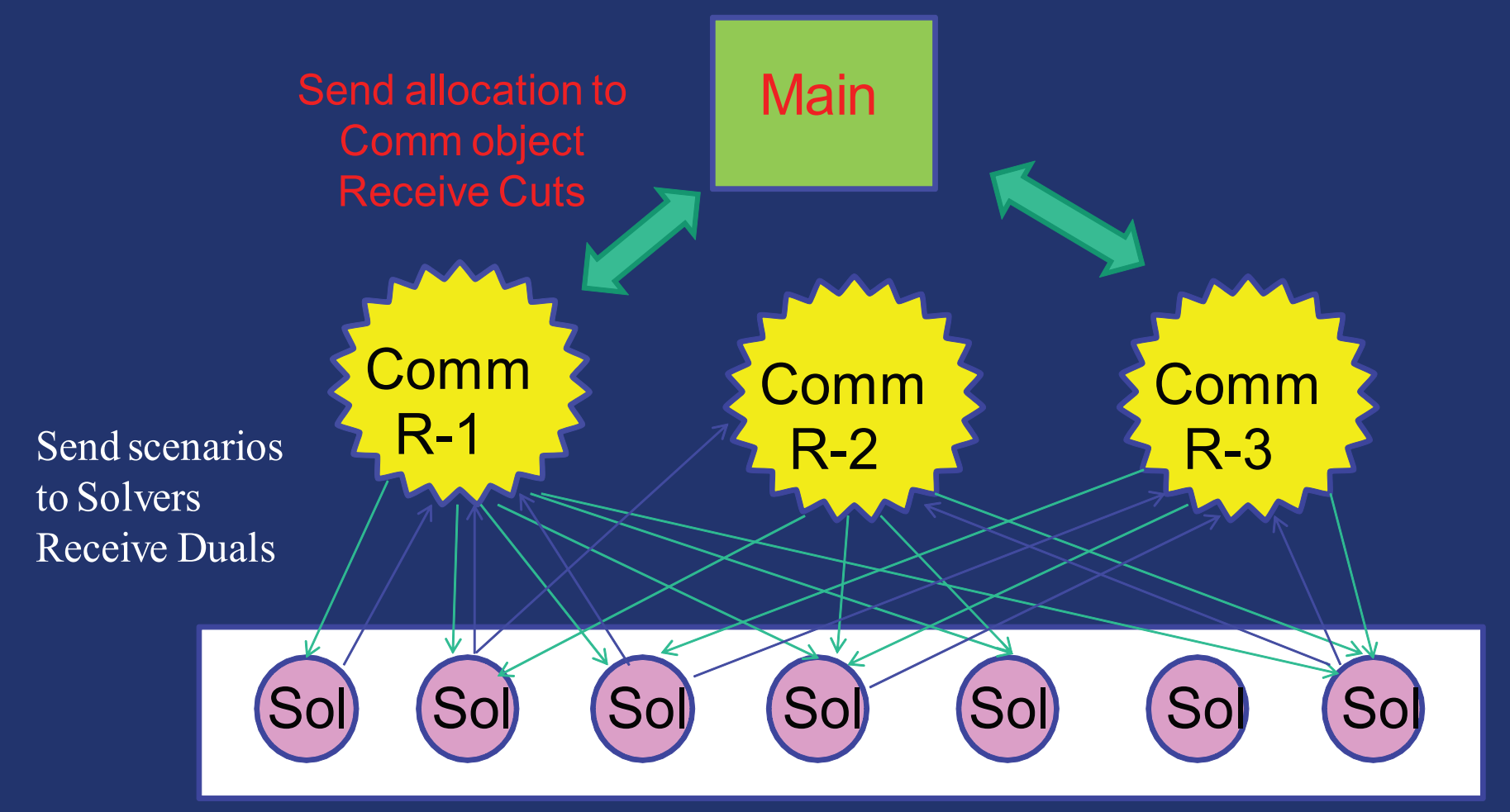
Solving the resulting Stochastic Program (Bender's *method)*

Master Worker Parallelism
- Single stage 1 compute object (master) proposes new allocations
- Collection of stage 2 compute objects (workers) provide feedback for each allocation
- Multiple rounds of master-worker interactions until optimal allocation is found

LP/IP using a numeric library: Challenges

Issue: Load imbalance because of variable, unpredictable compute grain sizes
Solution: Use a work request mechanism instead of a priori load distribution

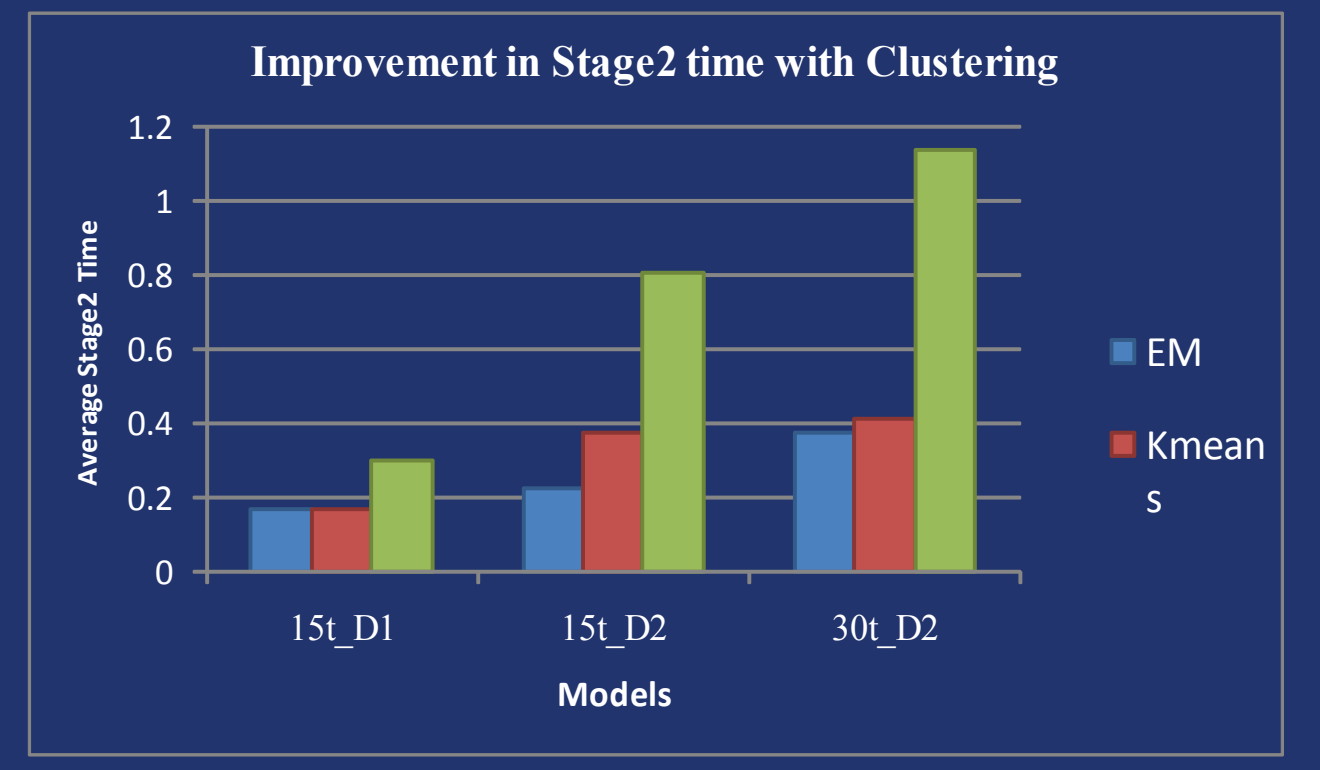
- Stage 2 objects request work when idle
- Orchestration object balances load by responding to stage 2 work requests
- Stage 2 objects keep working until allocation is evaluated under all scenarios


Send allocation to Comm object Receive Cuts
Main
Comm R-1
Comm R-2
Comm R-3
Send scenarios to Solvers Receive Duals
Sol
Sol
Sol
Sol
Sol
Sol
Sol

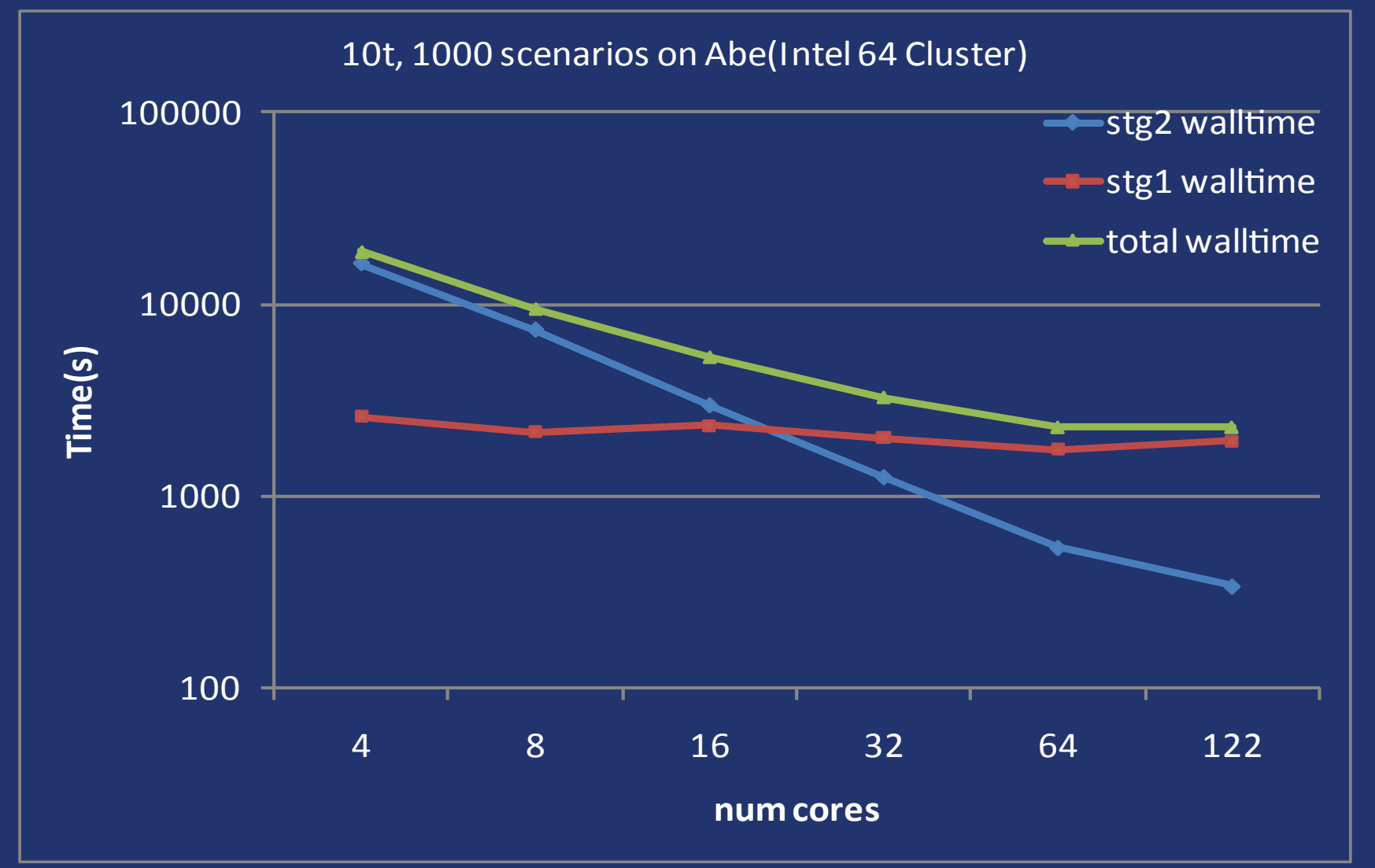
Issue: Internal Library state affects performance of future solves
Solution: Cluster similar scenarios to minimize solve times for the whole cluster

Scenario Clustering

Dependence between various stage2 scenarios.
Scenarios optimization starts from optimal dual basis of the last scenario solved
Solving similar scenarios successively (by forming clusters of scenarios) significantly reduces the stage2 solve time.



Clustering of scenarios gives significant reduction in stage 2 solve times

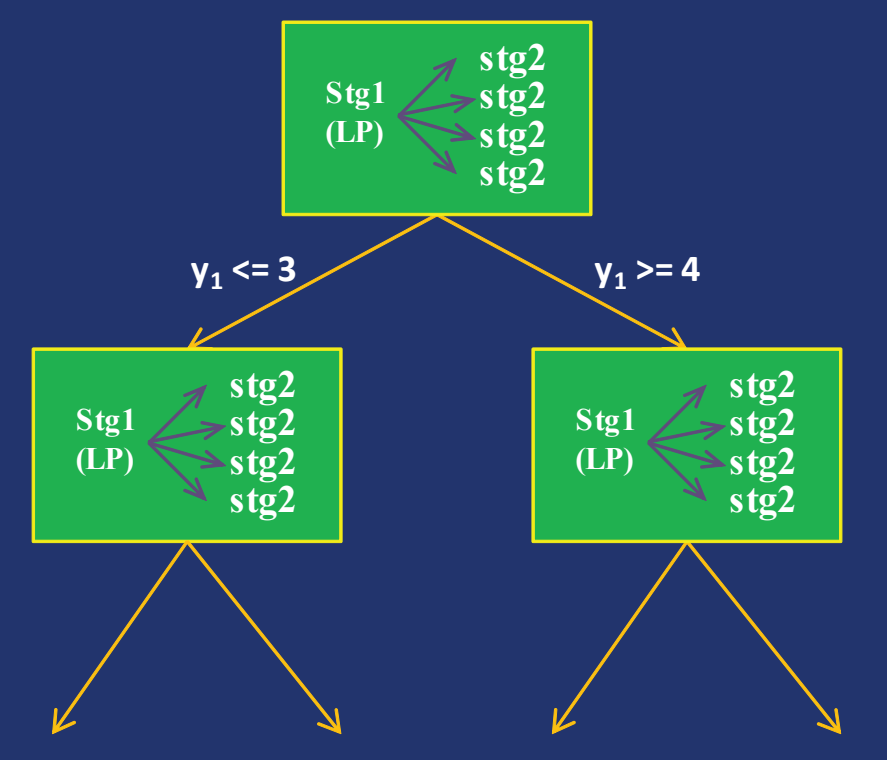


Amdahl's Law: Scalability plot of the Master-Worker parallelization based implementation with cut-management in stage1 and scenario based clustering in stage2

Branch and Bound Parallelism

Stage1 variables must be integers. Large solve times for stage1 IP prohibits scaling beyond a point (as also seen in the stage1 LP case). Henceforth, we present a massively scalable branch-and-bound based design to solve stochastic programs with integer stage1 programs.

Stage1 solved as linear program with branching on fractional variables. Keep branching until an integer solution is found or branch is pruned because lower bound exceeded the best incumbent known so far.



Branch-and-bound tree structure for multistage stochastic programs with stage1 as Mixed Integer Program

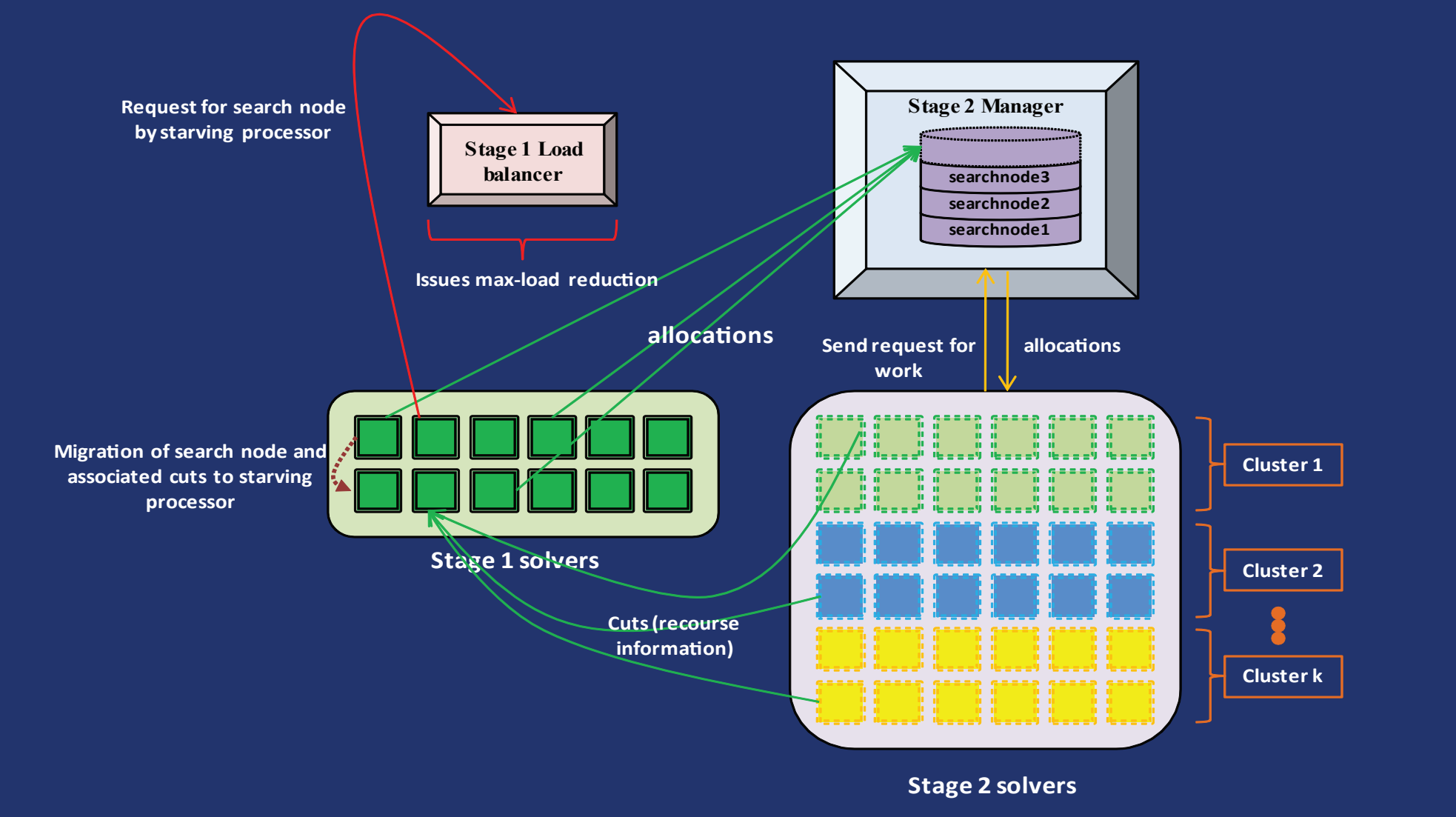
Lessons from Master-Worker parallelization and Motivations for the proposed design

1. Amount of computation per search node in branch-and-bound tree very large (could be hours for real world problems).
2. Finding single solution quickly more important than finding many solutions after a long period of time. Former helps pruning the tree and thus reduces the work.
3. Memory requirements per search node are very large because of accumulated recourse information from stage 2.
4. Highly variable stage1 and stage2 solve times call for load balancing and pull-based design for getting work.

Design Approach

- Categorize the available processors into two sets of stage1 and stage2 solvers. Each processor is associated with a solver (either stage1 or stage2).
- Stage1 solver maintains two queues - *ready* and *waiting*
  - *ready* queue - search nodes for which stage2 recourse information has been calculated (by stage2 solvers)
  - *waiting* queue - search nodes sent for stage2 scenario realizations
- Two dedicated processors act as stage1 load balancer and stage2 manager
- Periodic redistribution of processors between stage1 and stage2 solvers.
  reduces memory requirements per search node

However, it is important to maintain cut locality as cuts from one end of the tree not very useful for nodes in other end of tree



Overall design of the branch-and-bound strategy

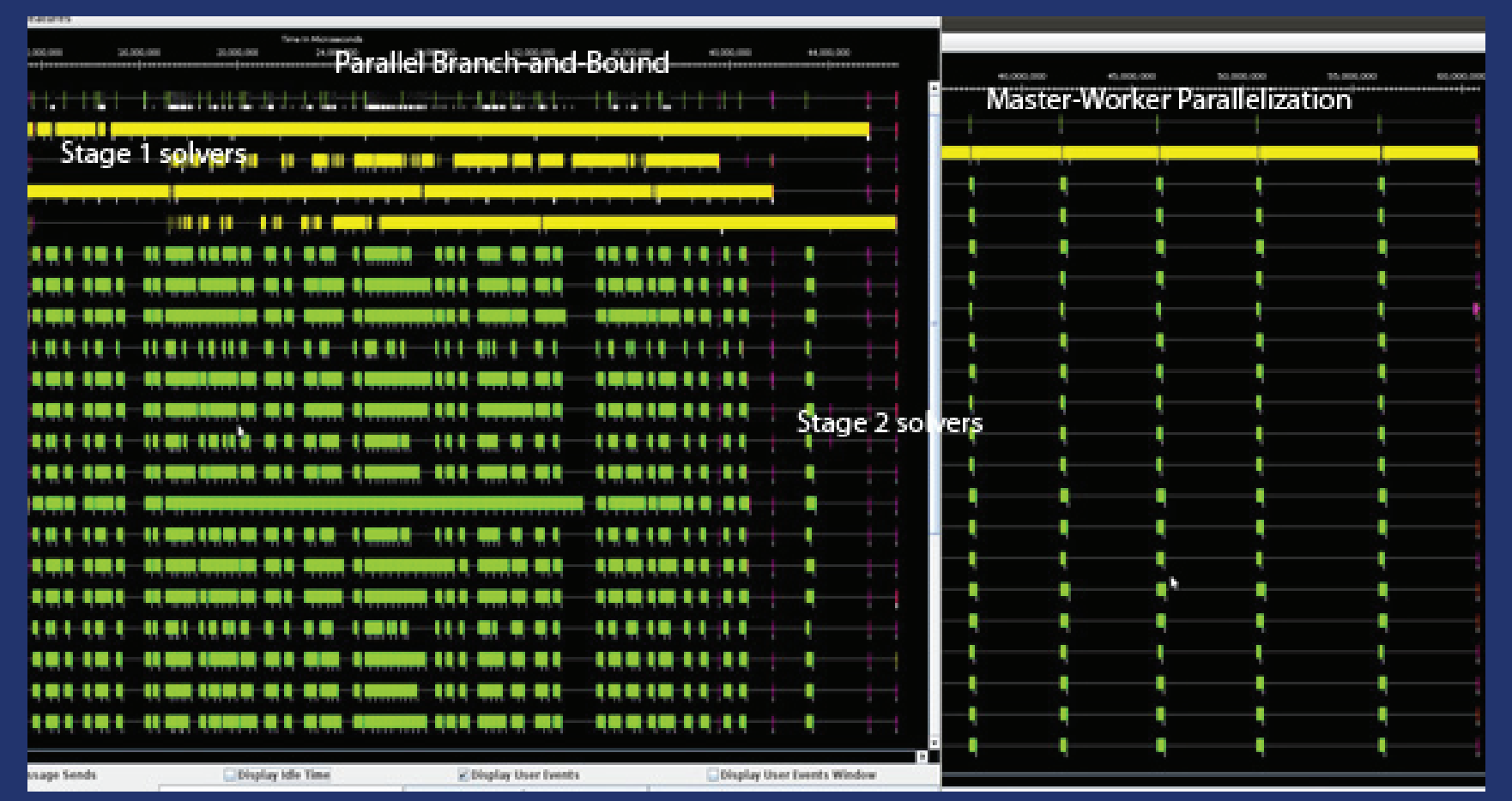
Workflow

- Start with X stage1 solvers and Y stage2 solvers
- Use adaptive convergence criteria to create enough search nodes in the beginning for idle stage1 solvers
- Same cuts used for all search nodes explored on a given stage1 solver (reduces memory requirements per search node).
- Single Stage1 solver responsible for exploring the entire subtree generated from a search node (unless there is work stealing)
  Ensures cut locality
- Combination of best-first and depth-first strategy used to explore the tree

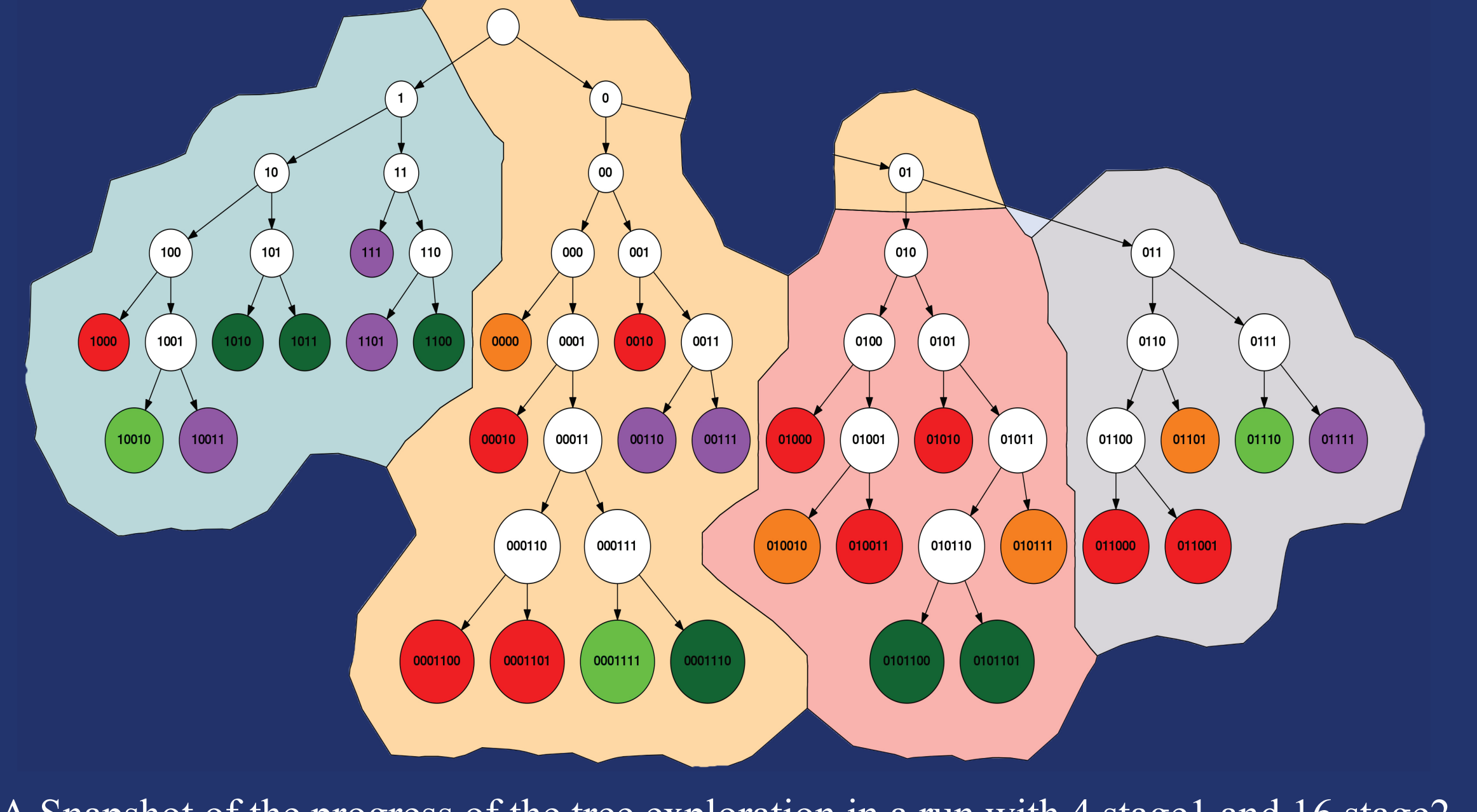
Periodic redistribution of processors between stage1 and stage2 solvers
- Increase number of stage1 solvers unless there is enough work to keep all stage2 solvers busy.
- Number of stage1 solvers should be such that all work generated by them gets instant stage2 response i.e. queue length of stage2 manager remains steady and close to zero. This keeps a check on the number of stage1 solvers.

Load Balancing amongst stage1 solvers
- When explored all nodes, request for work sent to stage1 load balancer which issues max-load reduction on stage1 solvers.
- Search node and associated cuts from the max-loaded solver sent to the requestor solver.

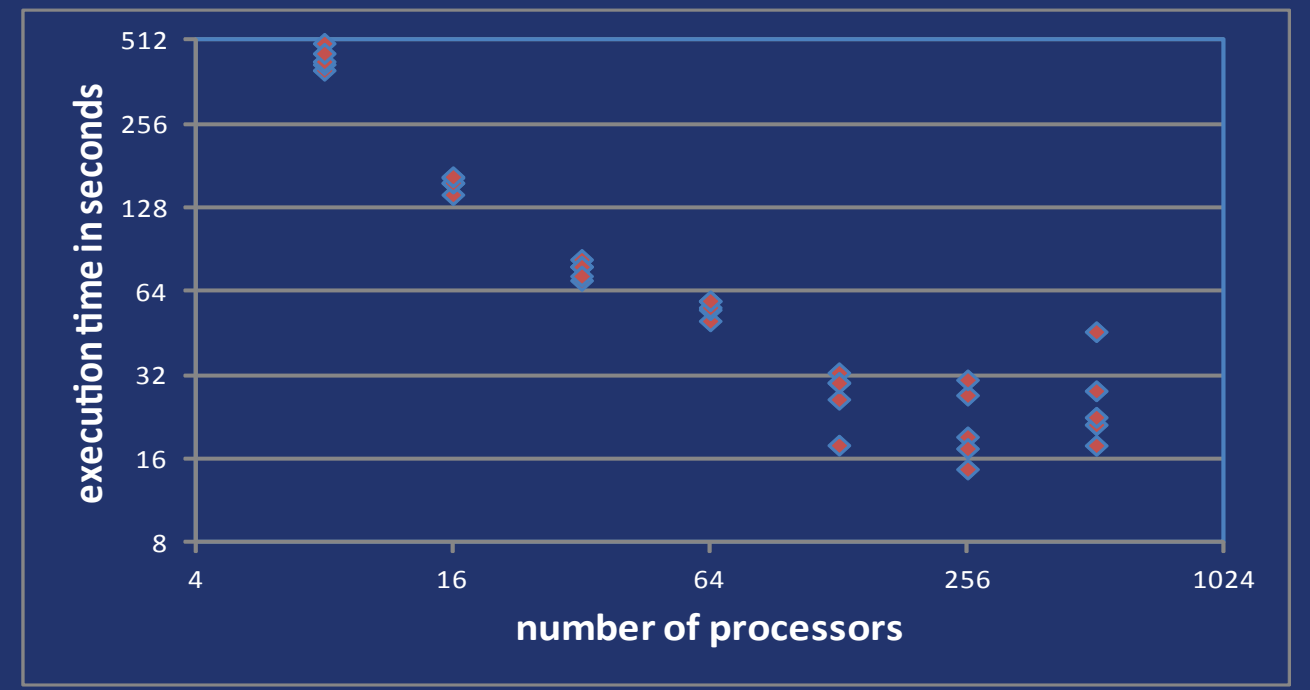
Results



Branch and Bound increases processor utilization and reduces the time to solution by more than half



A Snapshot of the progress of the tree exploration in a run with 4 stage1 and 16 stage2 solvers. Clusters represents the processor on which the search node was explored. Names of the nodes are their respective bit vectors based on the decision made on the parent node.
Search node color scheme- Red: pruned, Orange: incumbent integer solution, Green: currently being optimized in stage1, Blue: in waiting queue, Purple: in ready queue



Performance of the Parallel Branch-and-Bound Implementation on a Model with 225 integer variables in stage 1 and 120 scenarios in stage 2

Conclusion and Future Work

Intelligent search strategies along with design approaches to maximally harness the compute power of the high performance machines can be used to obtain optimal solutions for the stochastic problems which have remained unsolved in the past.

References

As a next step, we intend to incorporate into our design the ability to solve stage 2 problems also as integer programs.