

Using Multi-Core Systems for Rover Autonomy

Brad Clement

Team: Tara Estlin, Benjamin Bornstein, Brad Clement,
Paul Springer, Robert C. Anderson

Jet Propulsion Laboratory
California Institute of Technology

May 26, 2010



Task Objective

- Develop and demonstrate key capabilities for rover long-range science operations using multi-core computing
 - Adapt three rover technologies to execute on SOA multi-core processor
 - Illustrate performance improvements achieved
 - Demonstrate adapted capabilities with rover hardware
- Targeting three high-level autonomy technologies
 - Two for onboard data analysis
 - One for onboard command sequencing/planning
- Technologies identified as enabling for future missions
 - MSL, MAX-C, MSR rover missions
 - Titan aerobot/orbiter, flagship missions to Europa/Venus
- Benefits will be measured along several metrics:
 - Execution time / Power requirements
 - Number of data products processed per unit time
 - Solution quality



MER rover mission



MSR mission depiction



Selection of Technology

- Three high-level autonomy technologies:
 - Visual image analysis for rock identification (onboard MER)
 - Visual image analysis for terrain texture classification
 - Automated planning and scheduling for command sequencing (onboard EO1)
- Applicable to future missions and strong science support
 - Enable new/additional science to be collected
- All are computationally intensive
 - Rock identification takes > 15min on MER processor for single image
 - Texture analysis and planning have higher requirements
- Different strategies for parallelization
 - For rock identification, will split up image
 - For texture classification, will split up processing step
 - For automated planning, can split up among multiple spectrums (e.g., plan, search, quality improvements, etc.)

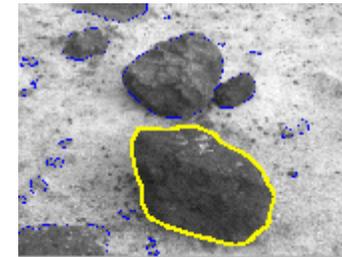


Image rock-finding on MER

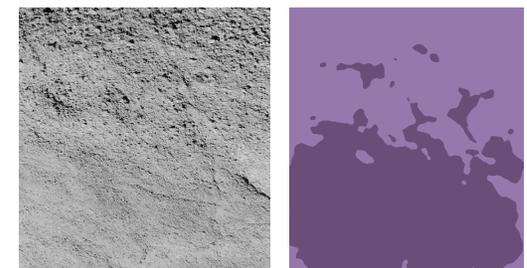
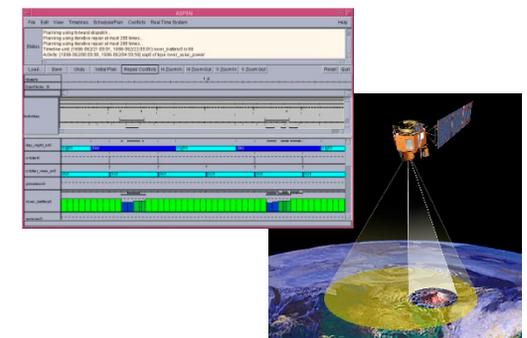


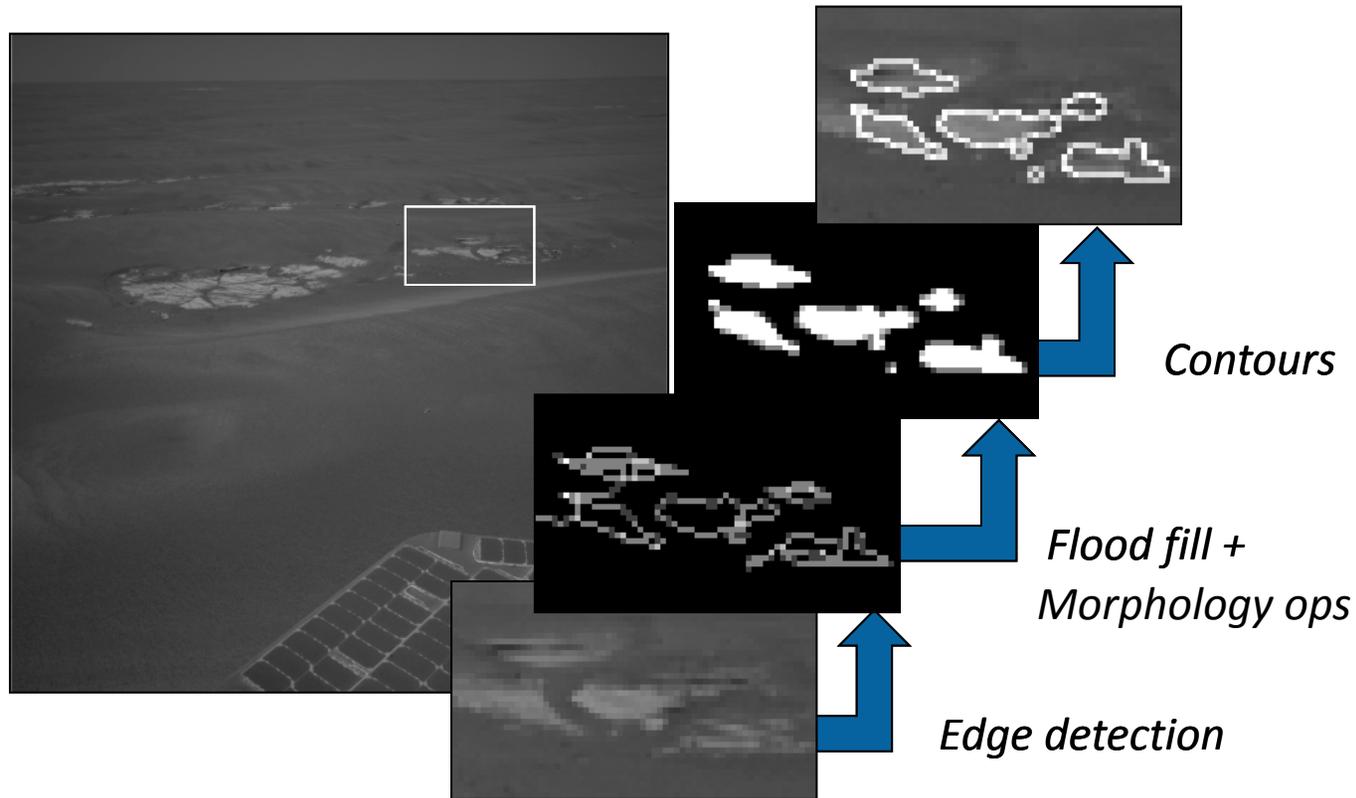
Image texture classification



CASPER on EO1



Image Analysis for Rock Detection



- Onboard analysis of rover visual images to automatically detect rocks and other terrain features
- Currently in use on MER Rover mission (RAD6000 platform)
- Time can vary dramatically based on number of edges in image



Rock Image Property Evaluation

Visual Texture

Variation in brightness at different orientations and spatial frequencies

Albedo

Average and standard deviation of image brightness for a rock

Size

From stereo, number of pixels or inscribed circle

Shape

- Eccentricity of ellipse fit to outline of rock
- Ellipse fit error
- Angularity

Smooth

Highly vesicular



Light

Dark



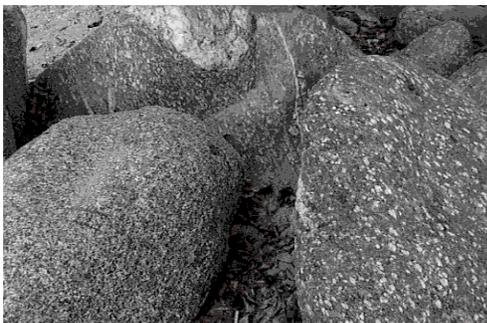
Rounded

Angular

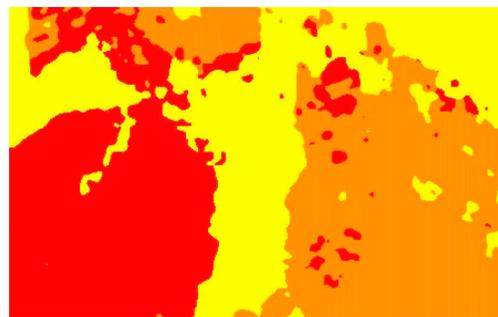


Image Texture Classification

- Goal: Separate image into homogeneous regions of similar texture
- Visual texture provides information about geologic texture
- Can be used to classify and/or detect rocks, detect layering, etc.
- Past approaches computationally intensive



*Igneous
rock*



*Metamorphic
rock*



Sedimentary rock





Image Texture Classification, cont.

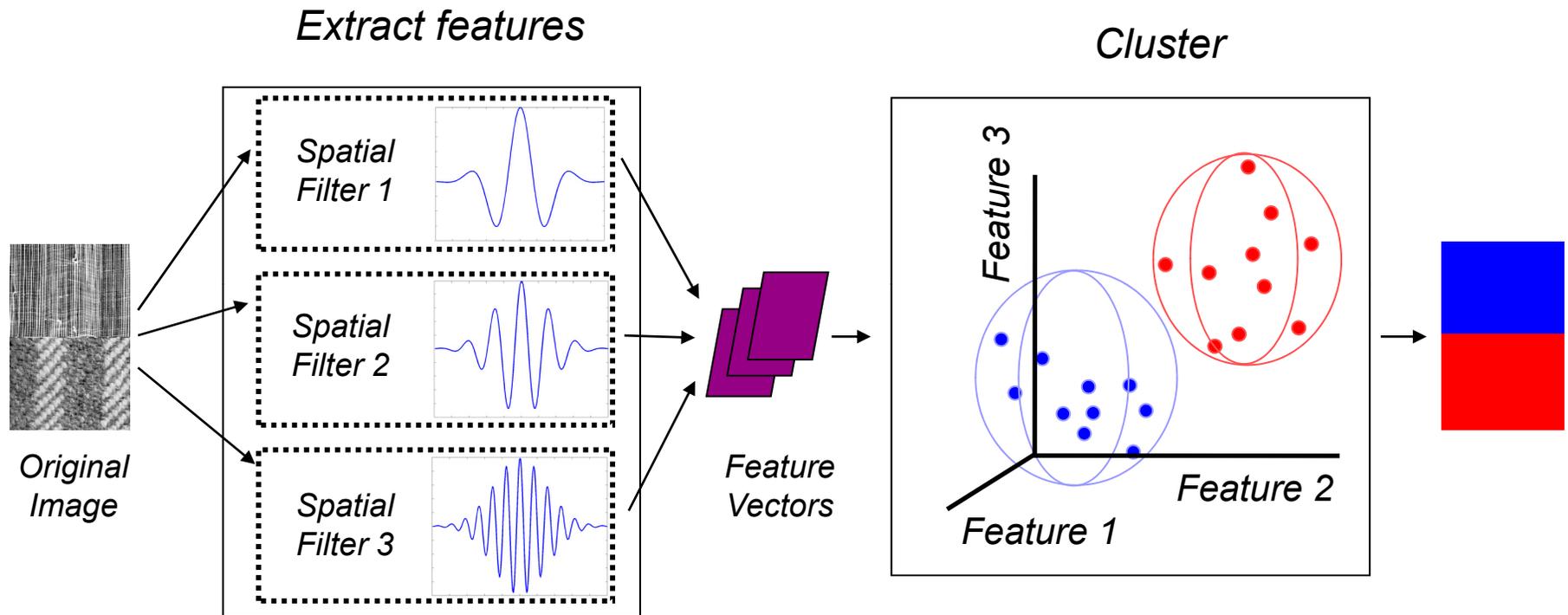


Image texture analysis ideal for adaption to multi-core processor

- Large bank of Gabor filters are applied to determine if each pixel matches different orientation and spatial frequencies
- Filters could easily be applied in parallel

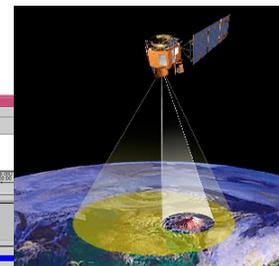
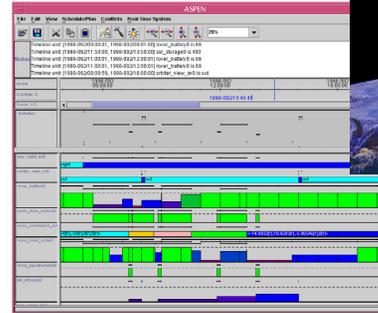
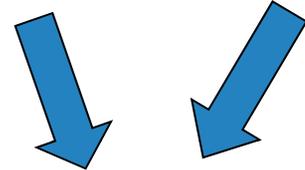


CASPER Automated Planning

- Enables automated modification or generation of robotic command sequences
- Approach:
 - Uses iterative-repair algorithm to find conflict-free activity plan
 - Uses model of activities, resources, states, etc.
 - Ensures that plan modifications do not violate resource or other operation constraints
- CASPER mission usage
 - Earth Observing 1 (EO1) satellite (2005-present)
 - Orbital Express mission (2007)
 - Modified Antarctica Mapping Mission (2000)
- Already have detailed model for research rovers

Goals:
*science requests,
downlink requests,
drive requests*

Constraints:
*memory,
power, time*



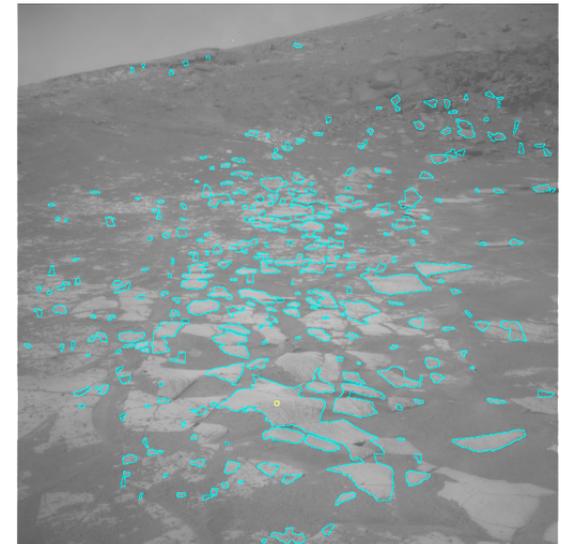
CASPER GUI

Command Sequence:
2003:233:16:49:57 CMD GO_TO_LOCATION (ROVER_X,...)
2003:233:17:56:57 CMD PAN_IMAGE(SITE_AZ,SITE_EL,...)
2003:233:18:07:06 CMD VISUAL_TRACK_TARGET(ROVER_X,...)
2003:233:18:07:06 CMD HAZARD_CAMERA_IMAGE(STEREO,...)
2003:233:18:07:16 CMD DRIVE(HEADING,DIST,EST_METHOD,..)



Technical progress: Rock detection

- Completed adaption of MER rock detection (Rockster) to Tileria Tile64 multi-core processor
 - Refactored Rockster to run standalone
 - Ported rockfinder and testing framework to Tile64
 - Created integer version of Rockster (no FPU)
- First performed simple evaluation of running multiple images on multiple cores
 - Runtimes scaled appropriately
- Designed and developed parallelization approach
 - Single image can now be automatically divided and processed on multiple cores
 - Can run on 1, 2, 4, 8, 16 or 32 cores
- Performed evaluation of rock detection on Tile 64 using set of MER Navigation camera images
 - Obtained **order of magnitude** speedup
 - Did observe degradation in quality of some results as increased number of cores





Rock Detection Parallelization

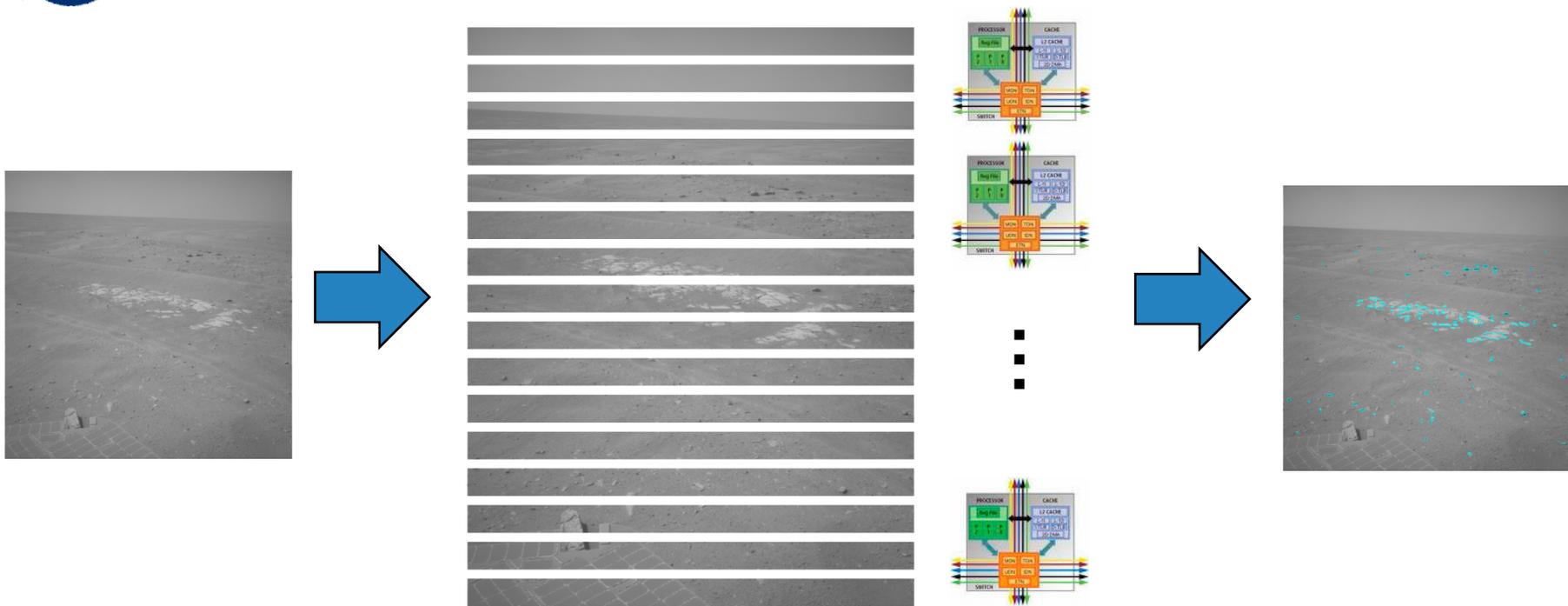


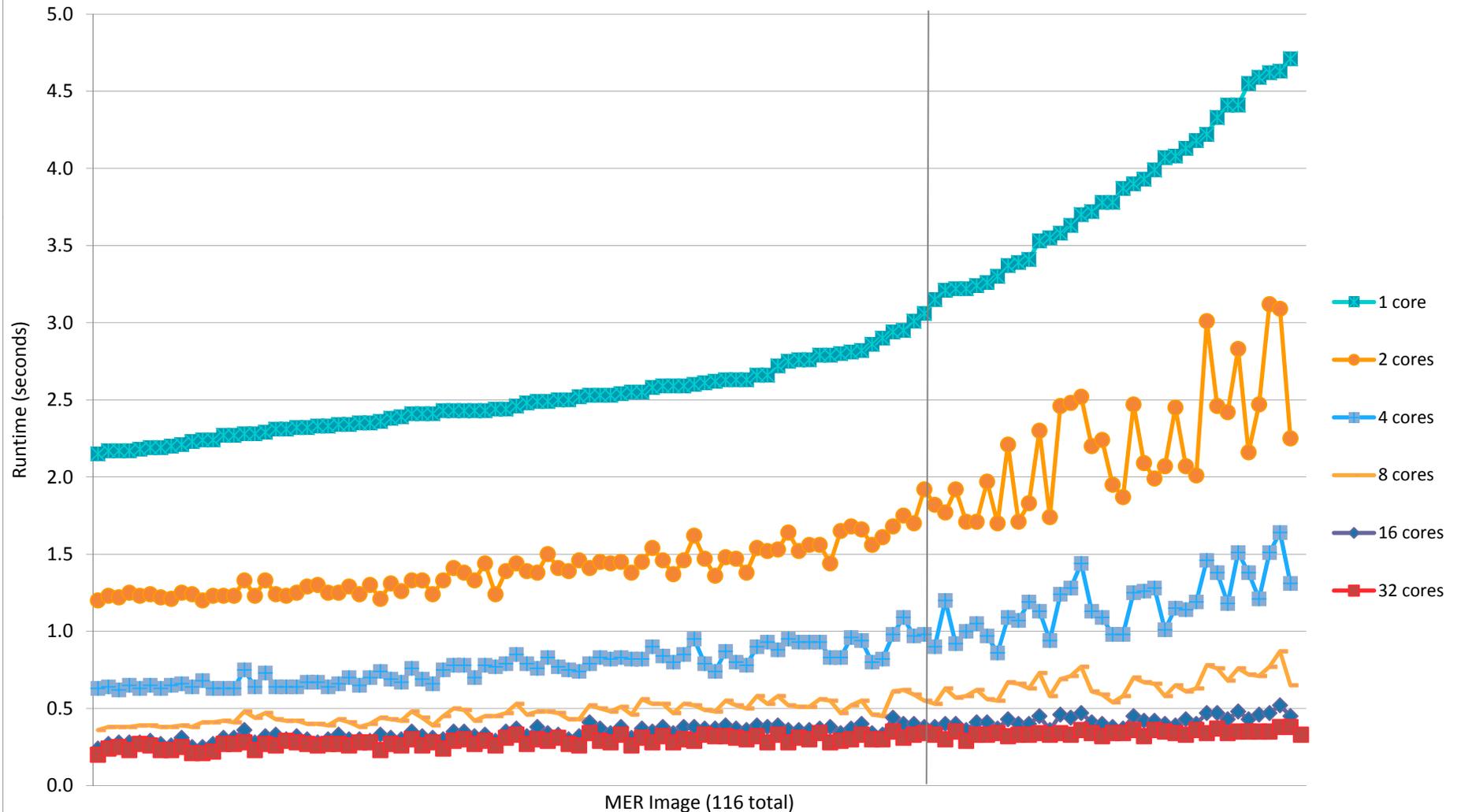
Image divided into strips that are processed for edges in parallel

- Allows speedup of single image processing
- Challenges include:
 - Selection of band size (tradeoff in solution quality vs. performance improvement)
 - Connecting contours across boundaries (may add later in year)
 - Strips vs. tiles
 - Max limit on number of rocks difficult to migrate evenly among cores



Results on MER Navigation Camera Images

Parallel Integer ROCKSTER Runtime on Tile64





Speedup Results for Increasing # of Cores

Jet Propulsion Laboratory

Number of Cores						
	1	2	4	8	16	32
Average speedup factor per image (vs. 1 core)	1X	1.8X	3.3X	5.5X	8.0X	9.7X
Average runtime over all 116 images	2.9 secs	1.6 secs	0.9 secs	0.5 secs	0.4 secs	0.3 secs

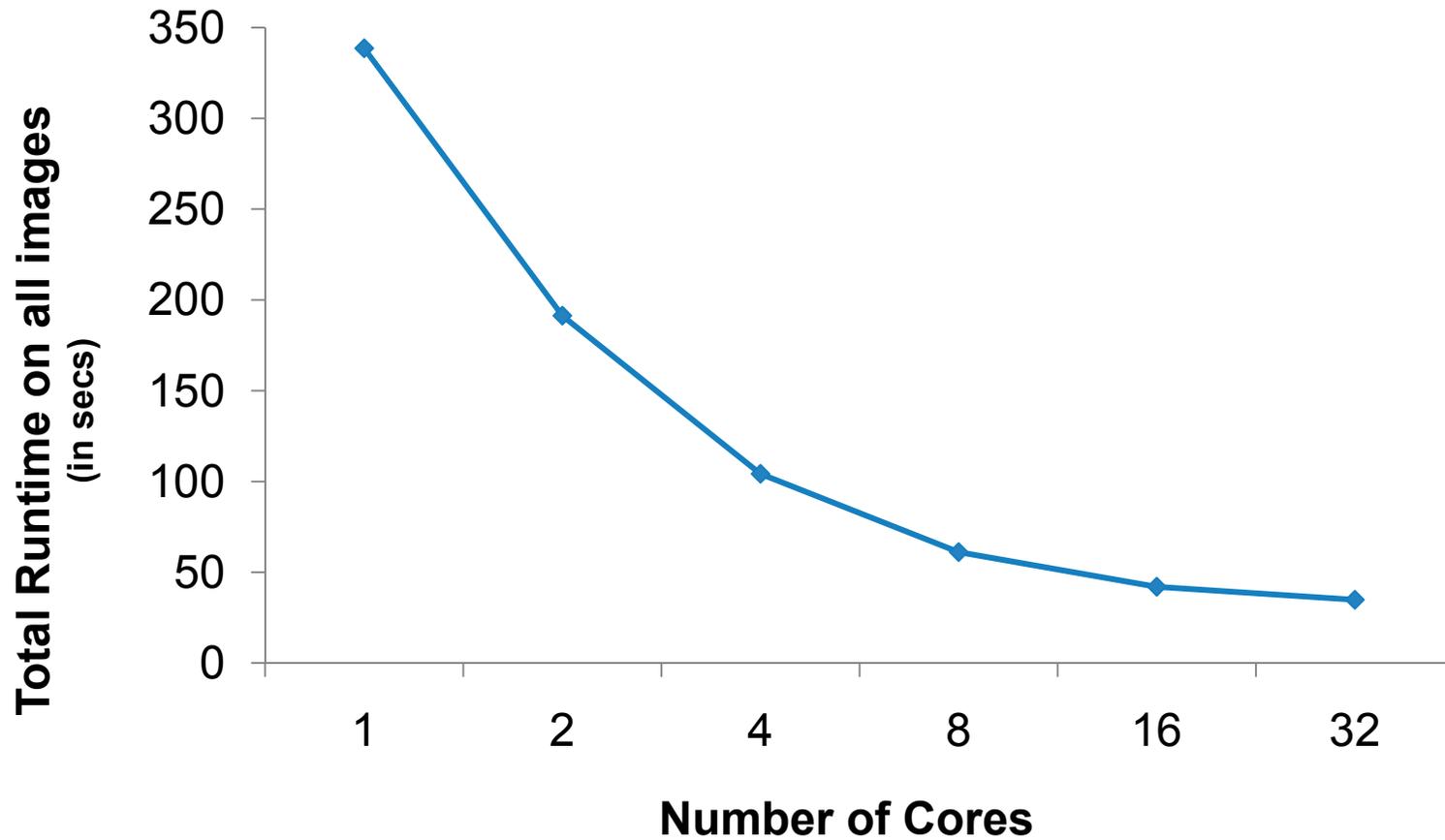
Minimum speedup on 32 cores (for single image): 7.4X

Maximum speedup on 32 cores (for single image): 14.5X



Scaling Results - Runtime

(Constant test set on differing # of cores)

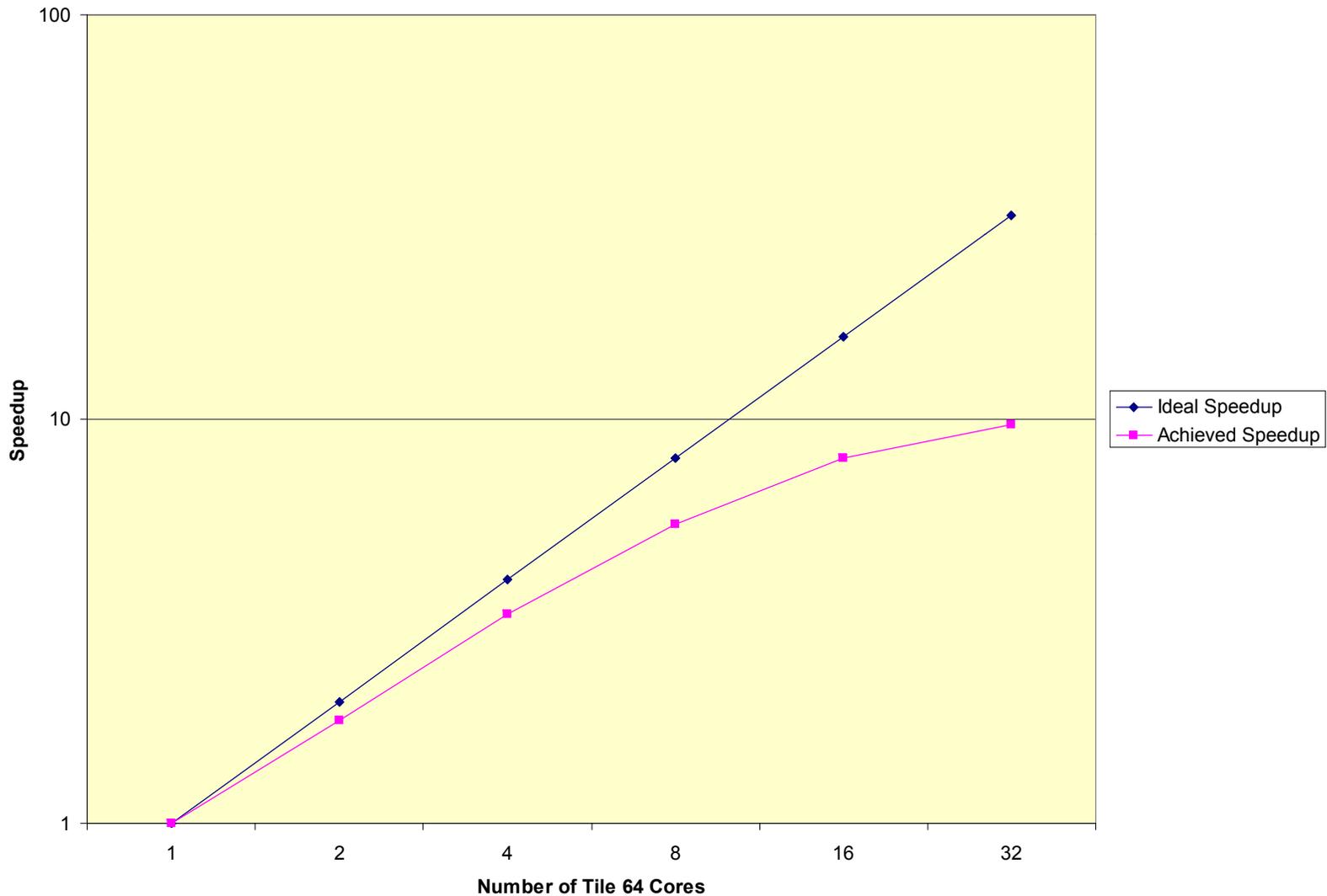




Scaling Results - Speedup

(Constant test set on differing # of cores)

Achieved vs Ideal Speedup



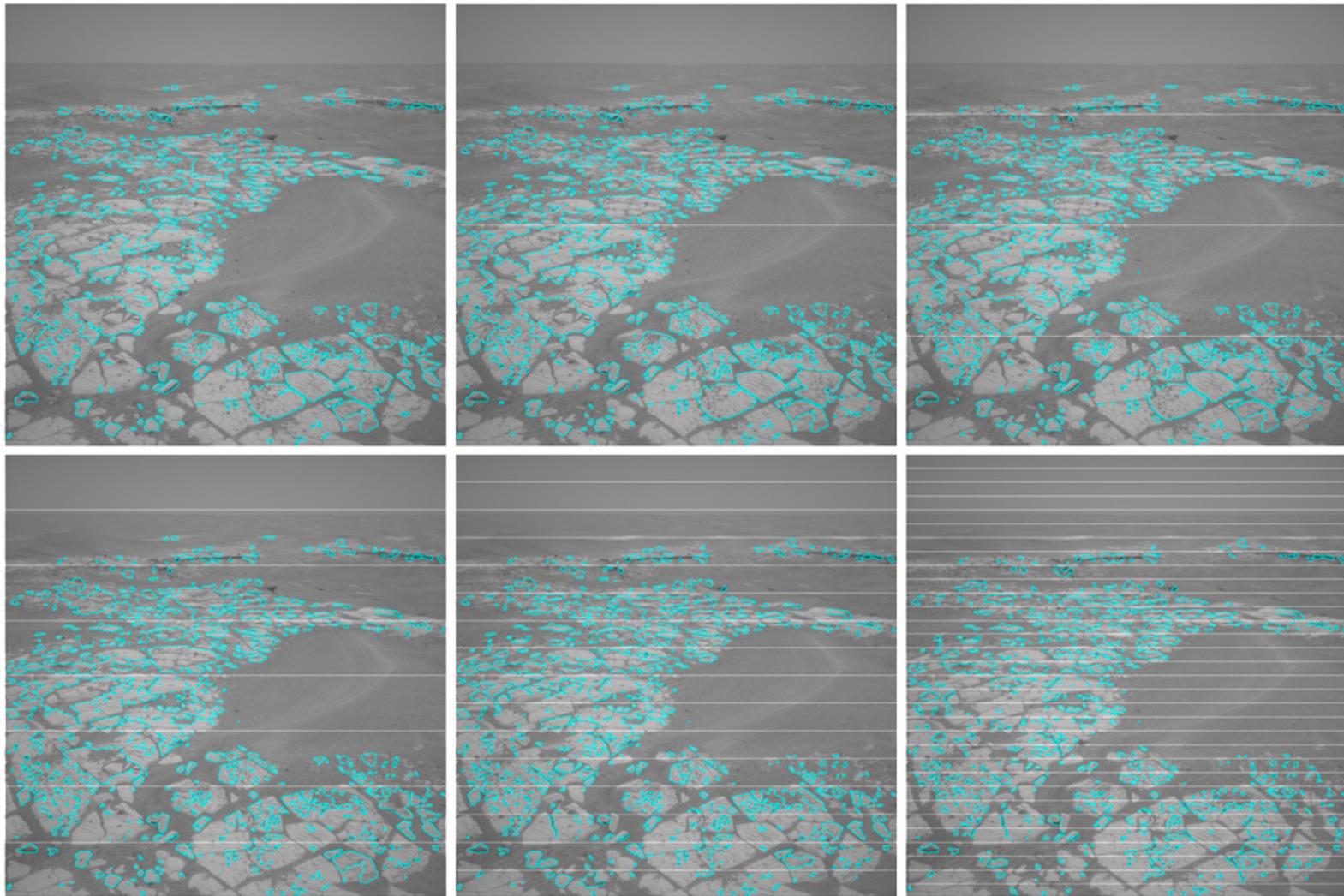


Evaluation Challenges

- Some degradation in quality when at 8 cores and greater, especially for larger rocks
 - Would like to investigate approach for connecting edges over boundaries
- Rockster internal rock cap of 255 rocks could affect runtime results
 - Each core used this cap independently
 - Difficult to fully remove cap; ideally would spread out among cores
 - Reran results with images that find < 255 rocks and got similar results
- Have not optimized use of caches
 - Peter Kogge has given us several good ideas
- All runs had full use of processor (100% CPU utilization)



Sample Result





Next Steps for Rock Detection

- Investigate optimal use of caches
- Investigate dividing image by tiling vs. strips
- Setup quantifiable quality evaluation
 - Could adapt past system used to evaluate performance of different rock finding approaches
- Improve scaling evaluation
 - Enable rock cap to scale appropriately
- Develop and implement approach for connecting edges between strip boundaries (probably not this year)
 - Complexity of this varies with different rockfinding algorithms
- Run using rover hardware (years 2 and 3)



MER Spirit rover image
from sol 810

Inputs to System Design and Fault Tolerance Tasks



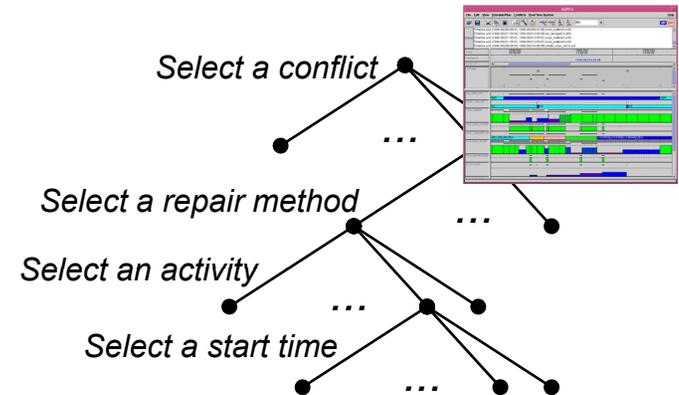
Jet Propulsion Laboratory

- Held series of meetings to provide inputs to System Design task on architecture qualities/requirements
 - Reviewed past ways targets applications have been fielded/integrated with other mission software
 - Outlined parallelization approaches for this work
 - Discussed design patterns that would be useful for this work (pipe and filter architecture, MapReduce)
- Supporting Fault Tolerance task
 - Provided application kernel (edge detection routine) to task
 - Supporting discussions on appropriate fault model / invariants for image processing
 - Supporting adding of assertions to application skernel



Technical progress: Automated Planning

- Investigating large list of potential ways for parallelizing CASPER
 - Potential for multi-core usage at multiple levels
 - Example options:
 - Split up computation / search process
 - Split up problem along timeline boundaries or activity/resource types
 - Split up memory / partial results storage
 - Use multiple cores for plan optimization strategies

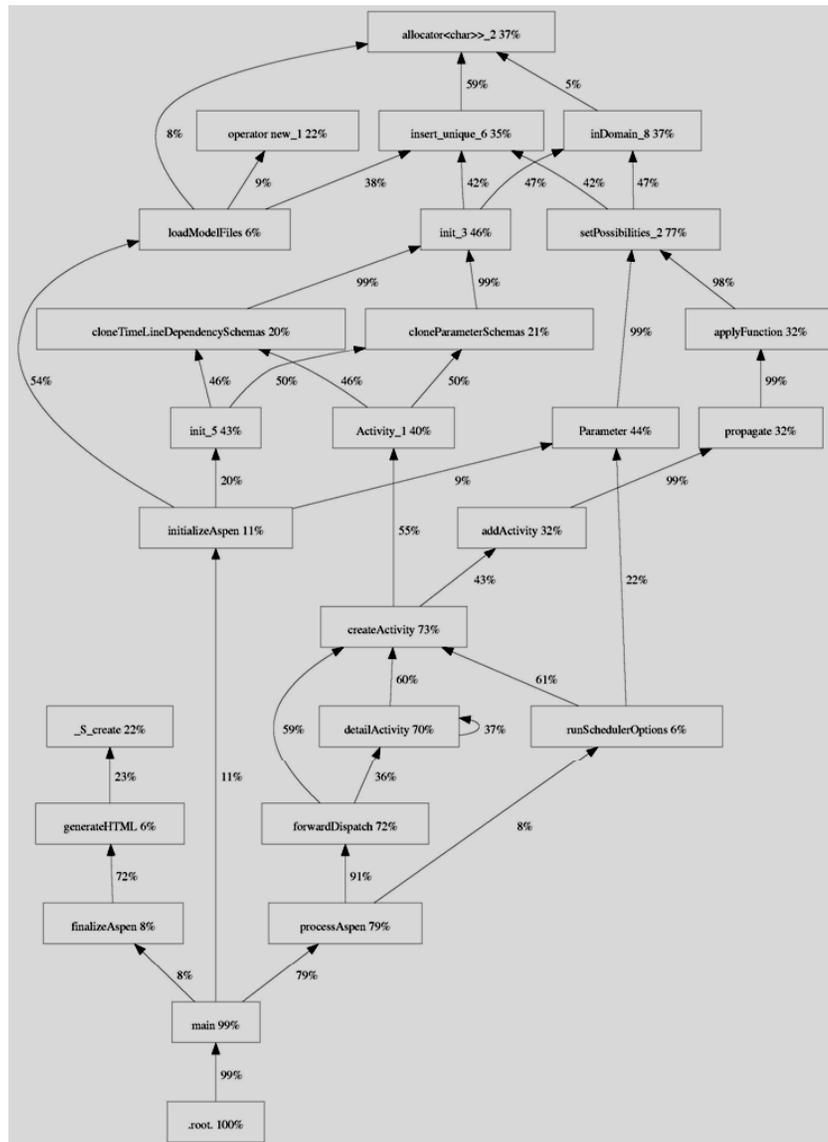


- Profiled CASPER on over 60 application models to determine bottlenecks and opportunities for parallelization
- Determined preliminary design for CASPER adaption to multi-core
- Wrote workshop paper at Int'l Planning conference on design options
- In process or porting CASPER to run on single Tile64 core



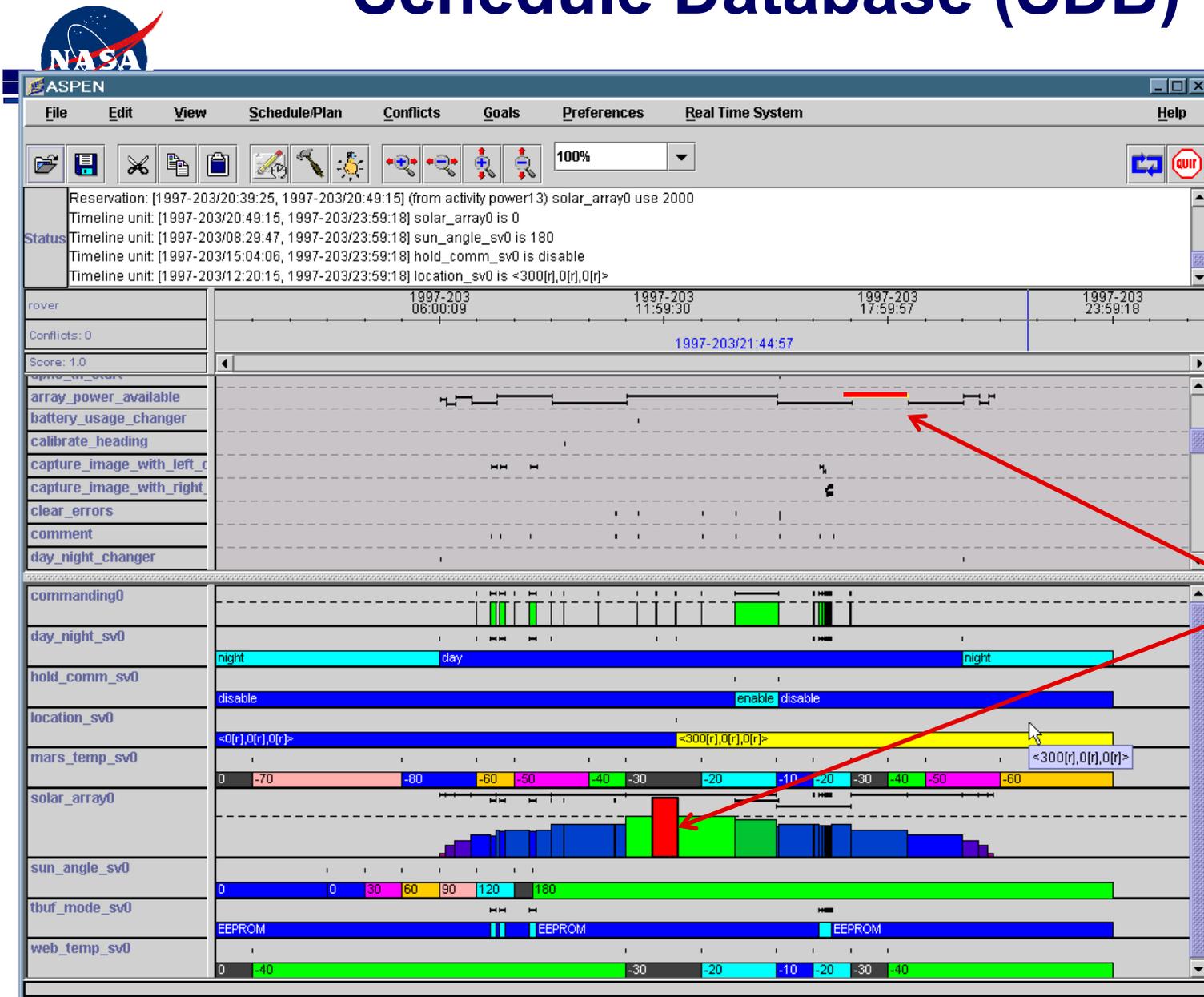


Technical progress: Automated Planning, cont.



- Computation time profile diagram for Earth-Observing One (E01) spacecraft model
- »
- Collected using CASPER regression test set
- Run for over 60 application models

Schedule Database (SDB)



activities

conflicts

*resource
& state
timelines*

Casper data → functions

bottlenecks

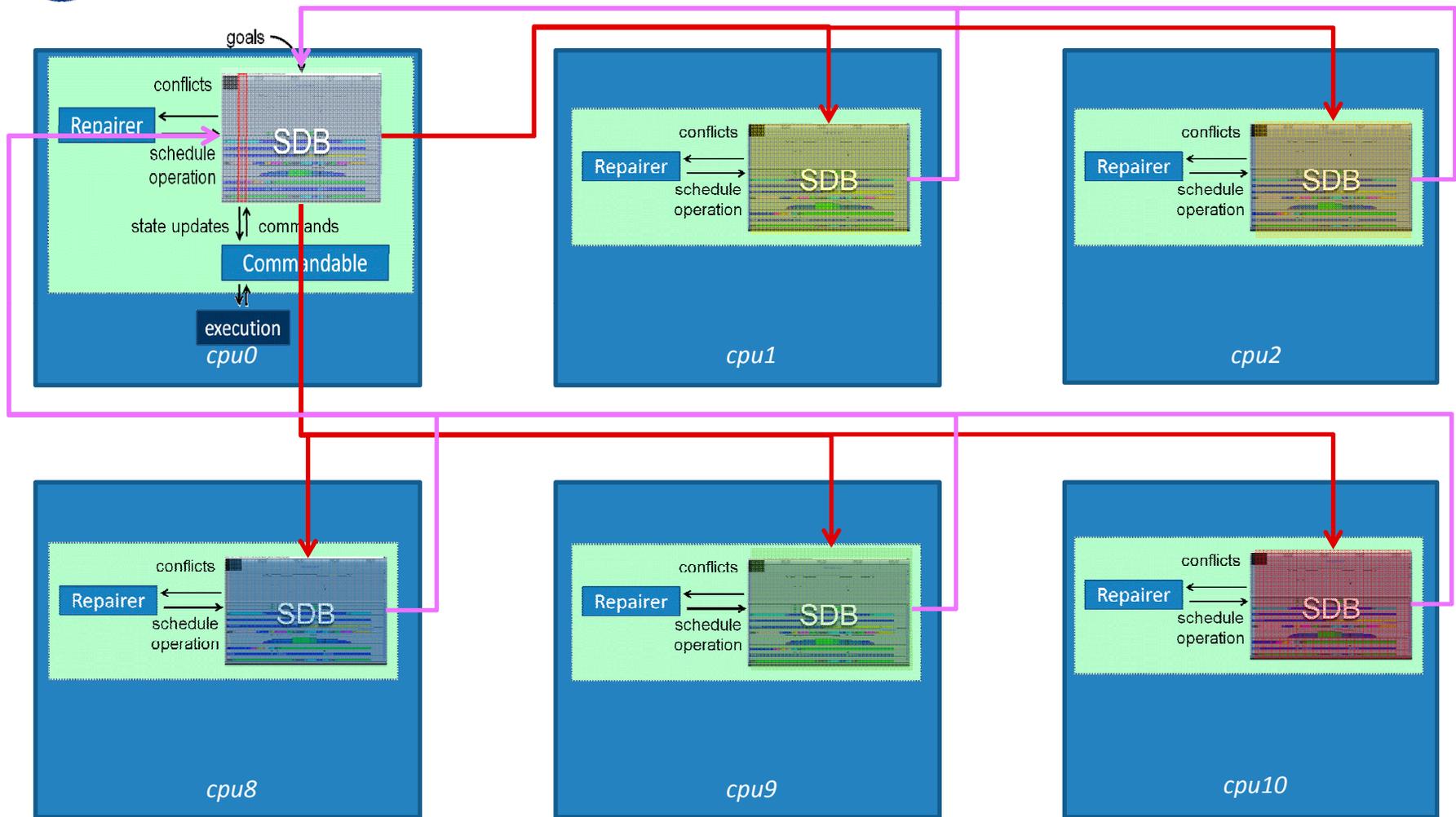


volatile objects	their functions
schedule database (SDB)	find, add, delete, schedule
<ul style="list-style-type: none">• activities<ul style="list-style-type: none">• valid intervals• parameters• dependencies, conflicts<ul style="list-style-type: none">• parameter constraint network (PCN)• temporal constraints, conflicts<ul style="list-style-type: none">• temporal constraint network (TCN)• reservations, conflicts	add, delete, move, place, lift, detail , abstract valid intervals get, set valid functional intervals, get conflicts apply function, propagate valid temporal intervals, get conflicts propagate connect, disconnect
<ul style="list-style-type: none">• timelines (state/ resource variables)<ul style="list-style-type: none">• timeline units (time → value)• reservations	valid timeline intervals , get conflicts compute, propagate get contributors
<ul style="list-style-type: none">• conflicts (type, rule, time)	get contributors
<ul style="list-style-type: none">• preferences (functions of SDB)	score
repairer/optimizer	repair/optimize
<ul style="list-style-type: none">• conflicts • valid intervals	gather conflicts choose conflict/preference choose method (e.g. move, add delete) valid intervals
commandable (commands, state updates)	command, update

parallel stochastic search, copied memory (central coordinator)



Jet Propulsion Laboratory

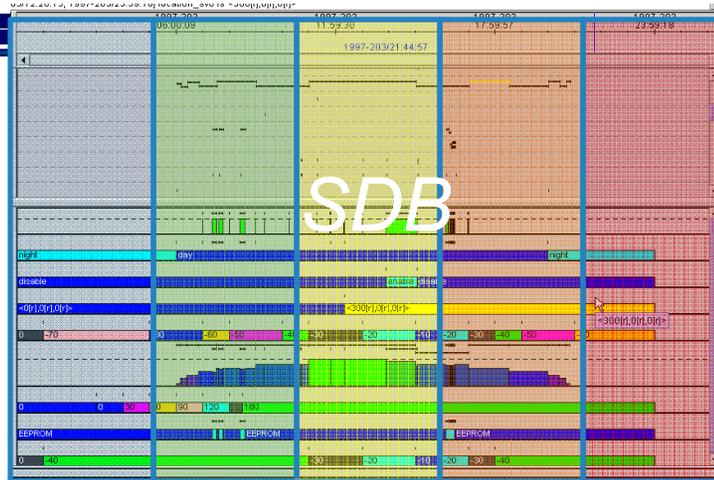


updated schedule
→
score, best schedule
←

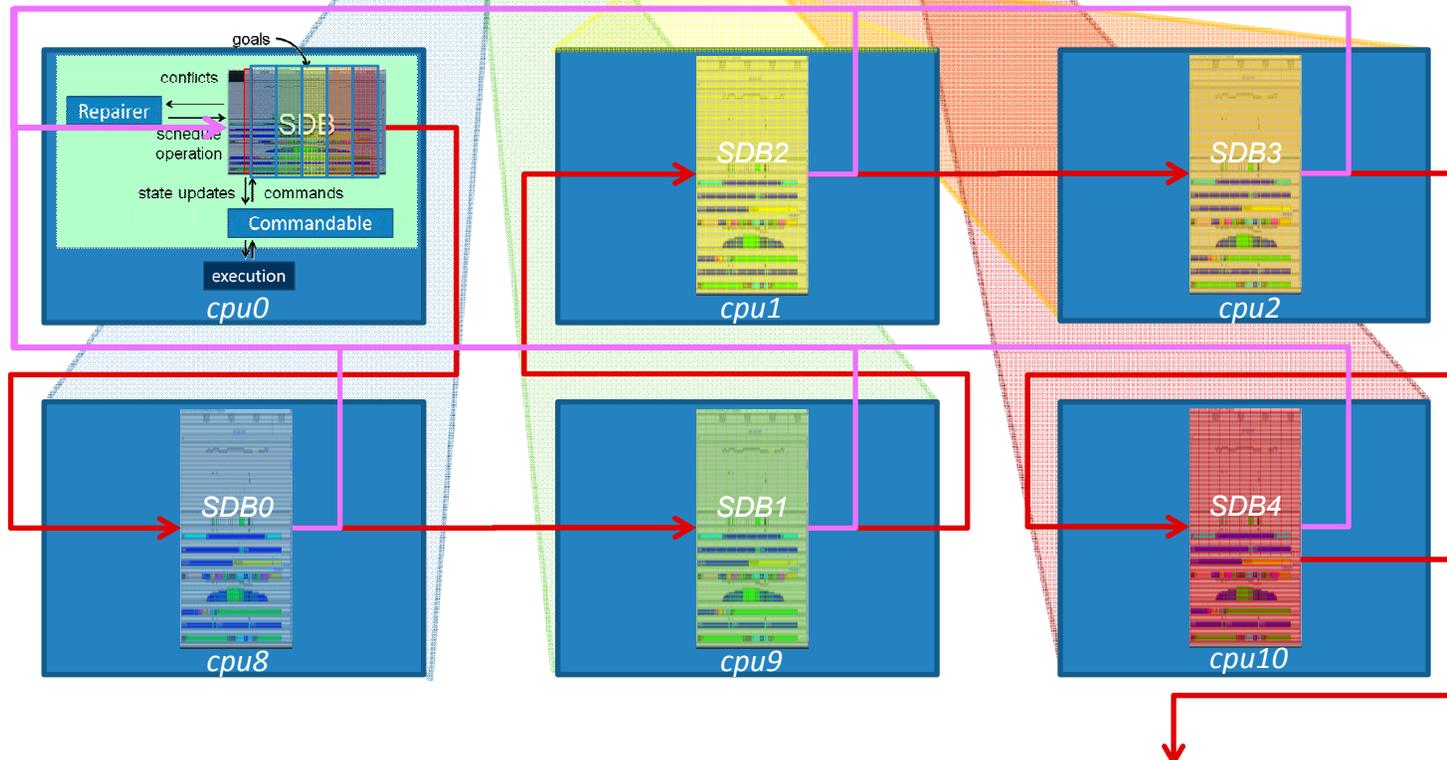
Parallel by time - activity and resources (master/slave coordination)



Jet Propulsion Laboratory



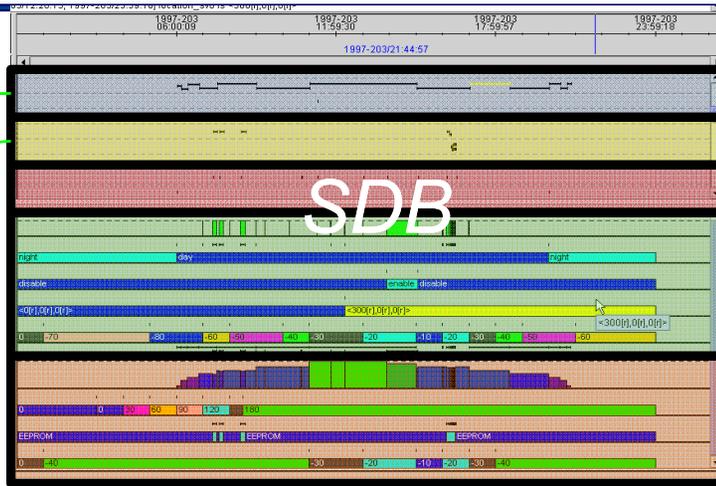
valid intervals, conflicts
activity Δs , propagation



Parallel by activity/timeline type (master/slave coordination)

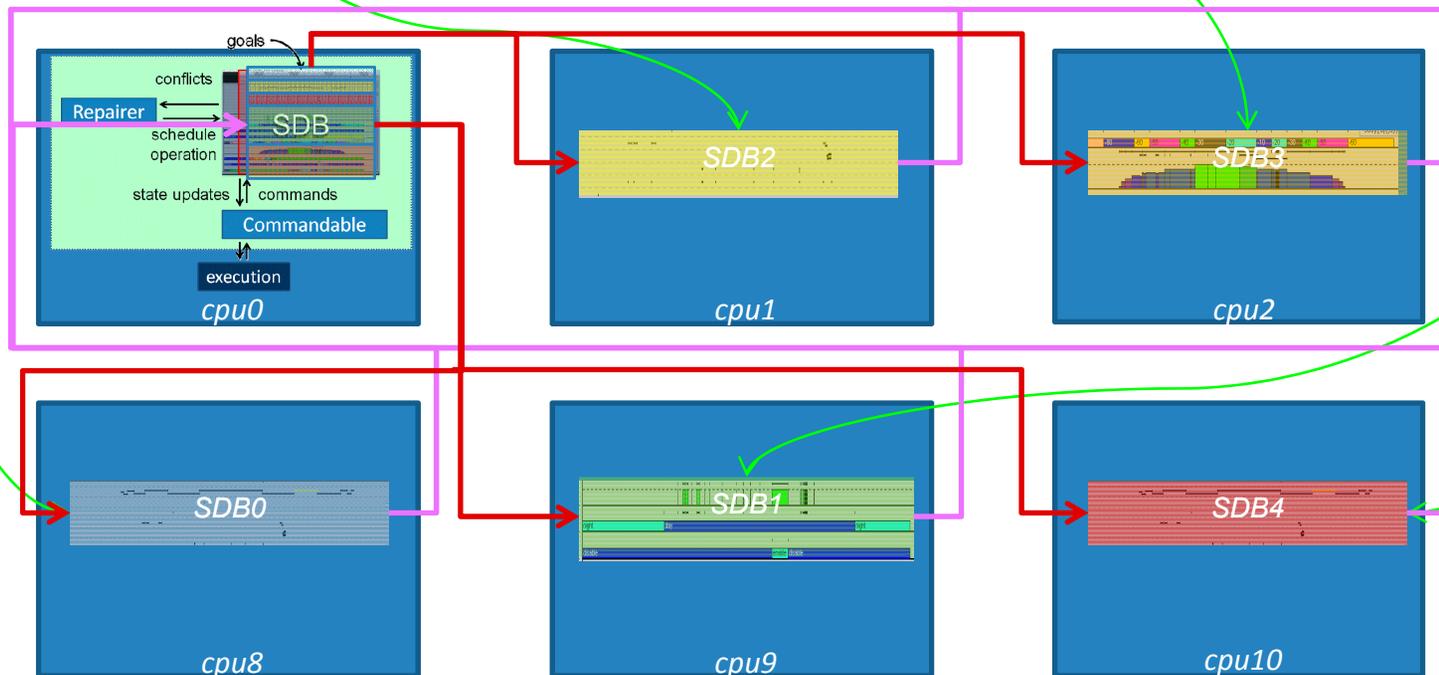


Jet Propulsion Laboratory



valid intervals, conflicts

activity Δ s, reservations





Challenges for Multi-core Adaptation

Jet Propulsion Laboratory

- No FPU on Tileria Tile64
 - FPU emulation but 10X slower
 - Creating integer versions of software is time consuming
 - Can effect quality of results.
- Designing parallelization approach
- Refactoring core algorithm to use this approach and communications software
- Platform / tools learning curve
- Slow I/O interface with Tileria board made running tests and profiling time-consuming



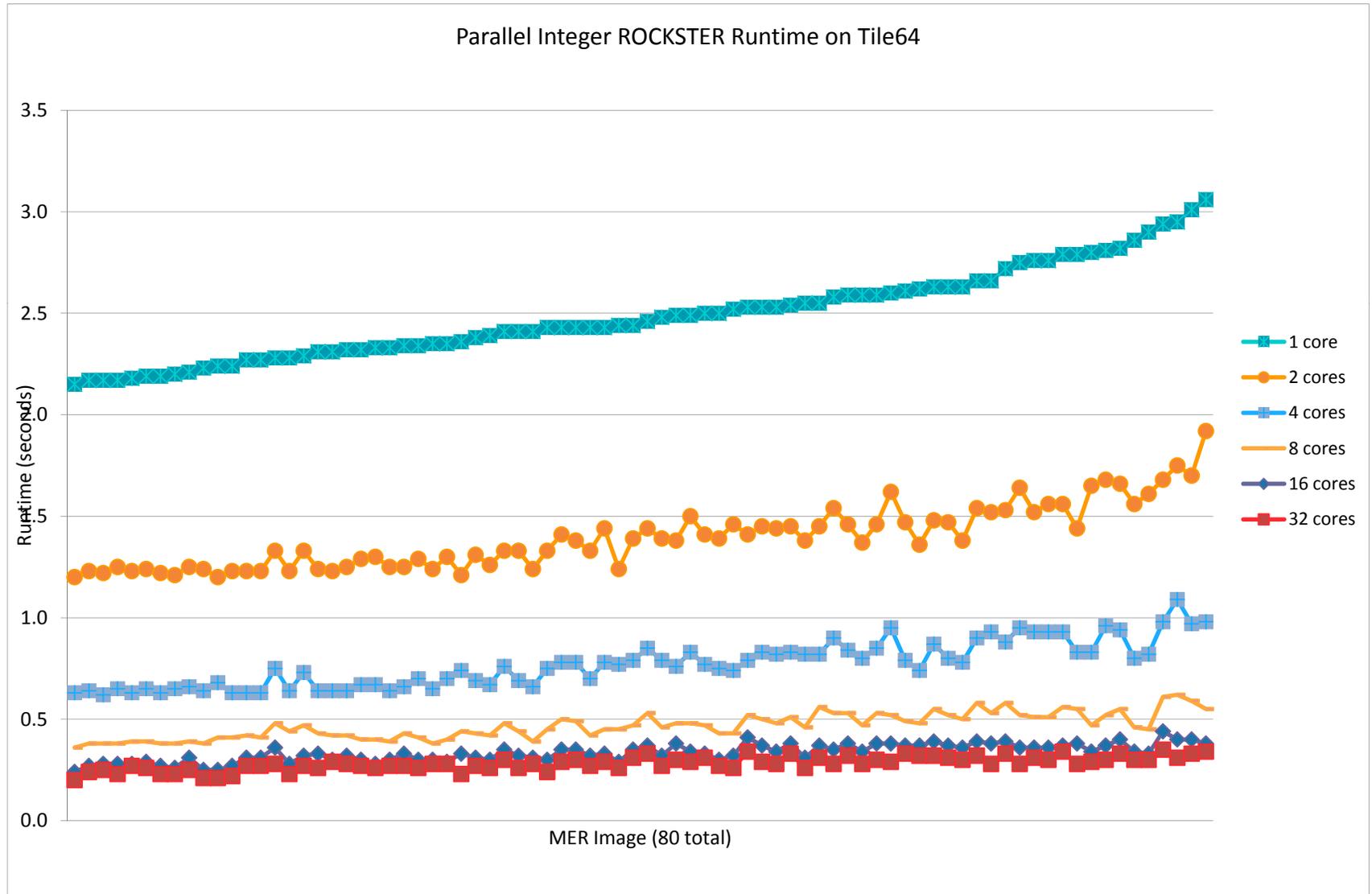
Where are we going?

- Room for autonomy software!
- Software architecture to support a wide variety of parallelization options
- Need to automate parallelization design
- Model processors
- Model software
- Use automated planning for allocating resources
- Automated planning/execution for real-time resource allocation
- → Automated parallelization for different instances of architectures, software, and data.

Extra Slides



Results on MER Navigation Camera Images (subset of 80 images where rocks found always < 255)





Overall Performance Results

(subset of 80 images where rocks found always < 255)

Number of Cores						
	1	2	4	8	16	32
Average speedup factor per image (vs. 1 core)	1X	1.8X	3.3X	5.4X	7.5X	8.9X
Average runtime over all 116 images	2.5 secs	1.4 secs	0.8 secs	0.5 secs	0.4 secs	0.3 Secs

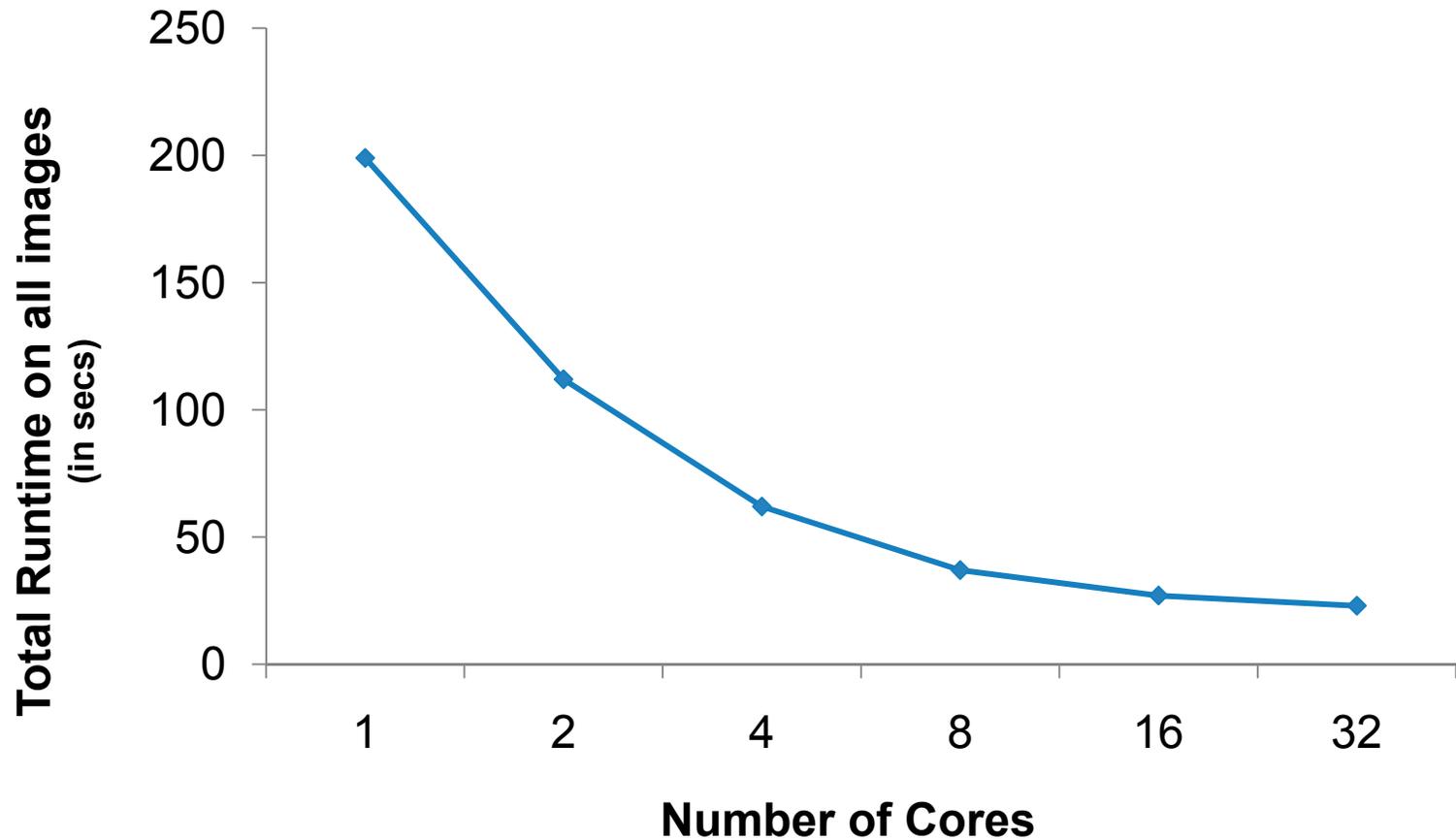
Minimum speedup on 32 cores (for single image): 7.4X

Maximum speedup on 32 cores (for single image): 10.8X



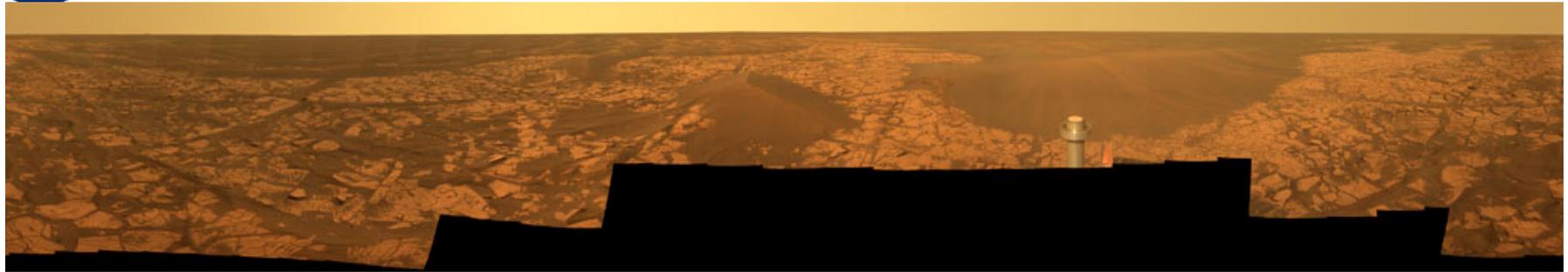
Scaling Results for 80 images

(Constant test set on differing # of cores)

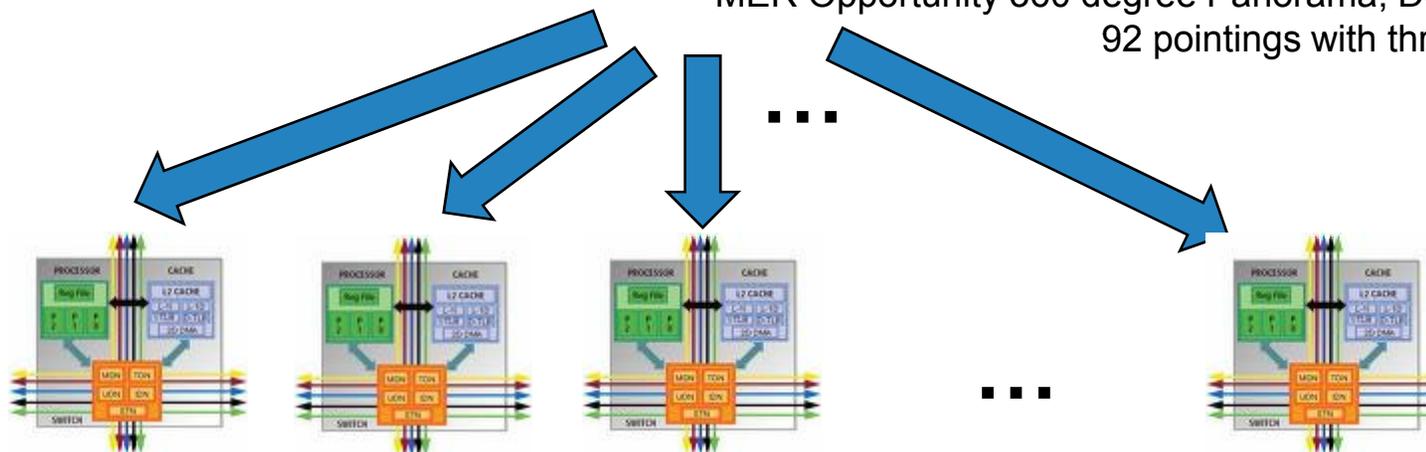




Simple Rockfinding Parallelization



MER Opportunity 360 degree Panorama, December 2008
92 pointings with three filters each



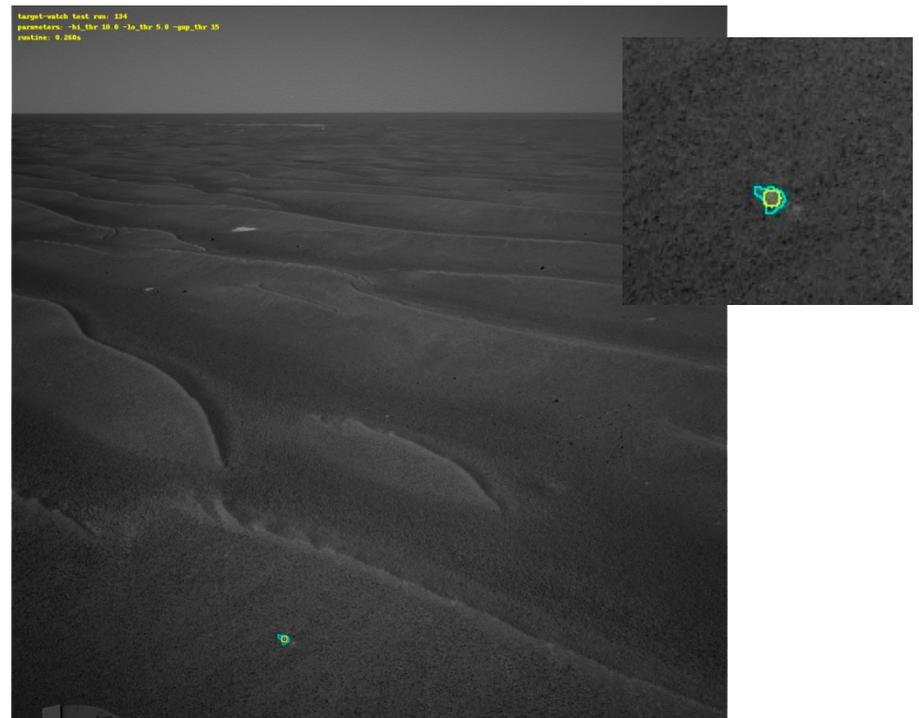
Individual images processed on separate cores

- Allows parallel processing of image panoramas (or large sets of images)
- Only minimal coordination required
- Performance improvement limited by image which is slowest to process
- Simple evaluation showed that runtimes scaled appropriately

Sample of AEGIS target selection on MER Navcams



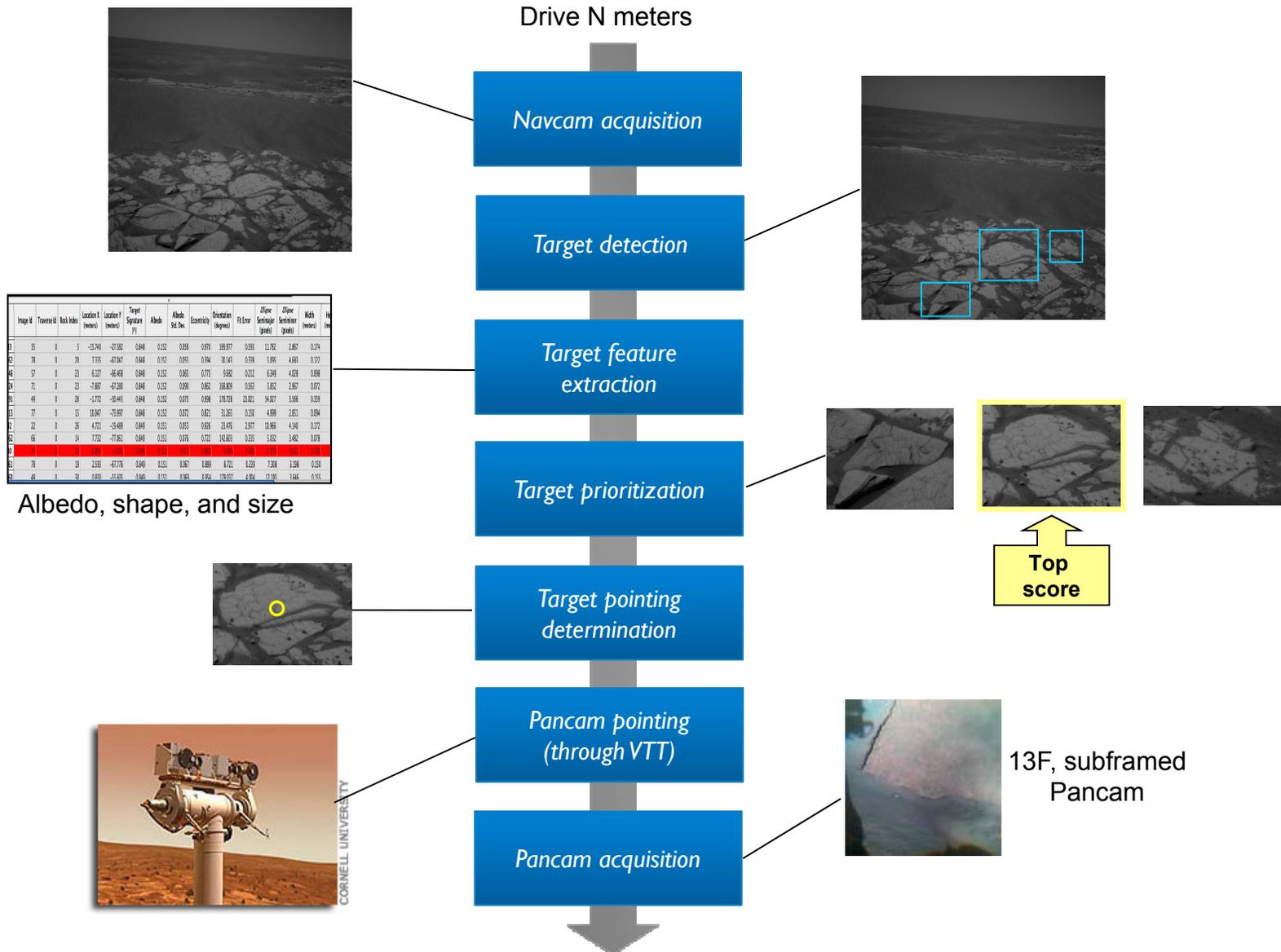
Jet Propulsion Laboratory



Top targets selected in two Navcam images. System was set to find the largest target.



AEGIS Process Example





Results on 120 MER Navigation camera images

Jet Propulsion Laboratory

	Number of Cores						Speed-up
	1	2	4	8	16	32	(1 core vs. 32 cores)
Sample image 1 runtime (in secs)	11.7	6.5	3.6	2.0	1.0	0.7	17X
Sample image 2 runtime (in secs)	49.3	27.9	12.6	4.9	1.8	1.1	47X
Average runtime over all images (in secs)	17.6	9.2	4.8	2.4	1.2	0.8	24X

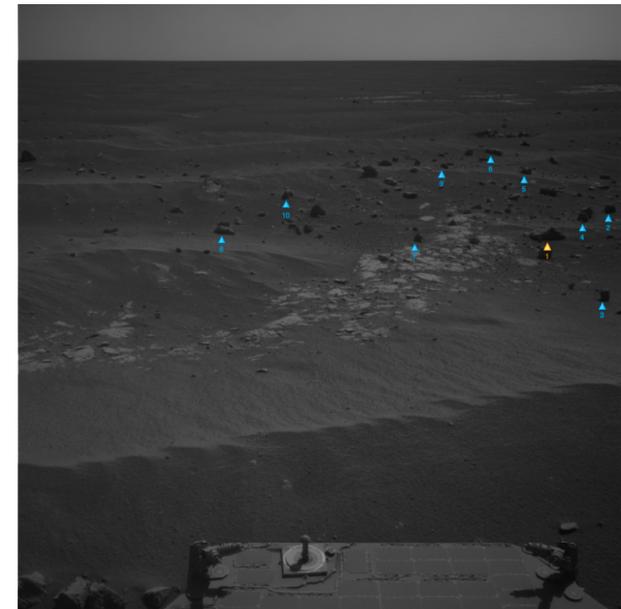
Things to note:

- All runs had full use of processor
- Each core capped at finding 255 rocks
 - Speedup could be higher if cap applied to all cores
 - Scaling difficult to evaluate
- Some degradation in quality when at 8 cores and greater



Image Analysis for Rock Finding, cont.

- Currently in use onboard the MER Rovers for automated target detection
- Critical element of onboard science, but very time intensive
- On MER rover platform (RAD6K) can require 15 mins or more of processing time
- Time can vary dramatically based on number of edges in image

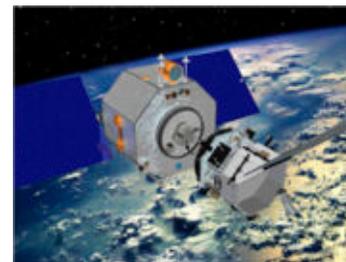


CASPER Applications and Mission Infusion



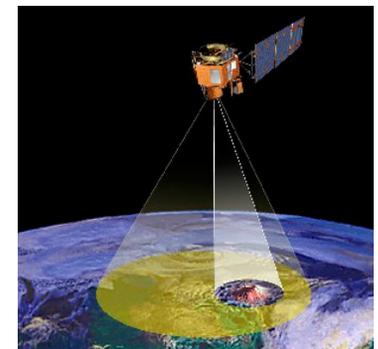
Jet Propulsion Laboratory

- Applications:
 - Opportunistic science
 - Respond to new science opportunities
 - Only add new science activities if resources allow
 - Plan optimization
 - Reason about task priorities, resource usage, constraints, long-term objectives, etc.
 - Can be used to take advantage of unexpected resource availability
 - Re-planning in response to problems
 - Resource or time oversubscription
 - Faults or unexpected states



Orbital Express

- CASPER mission usage
 - Earth Observing 1 (EO1) satellite (2005-present)
 - Orbital Express mission (2007)
 - Ocean Observatories Initiative AUVs (2009-present)

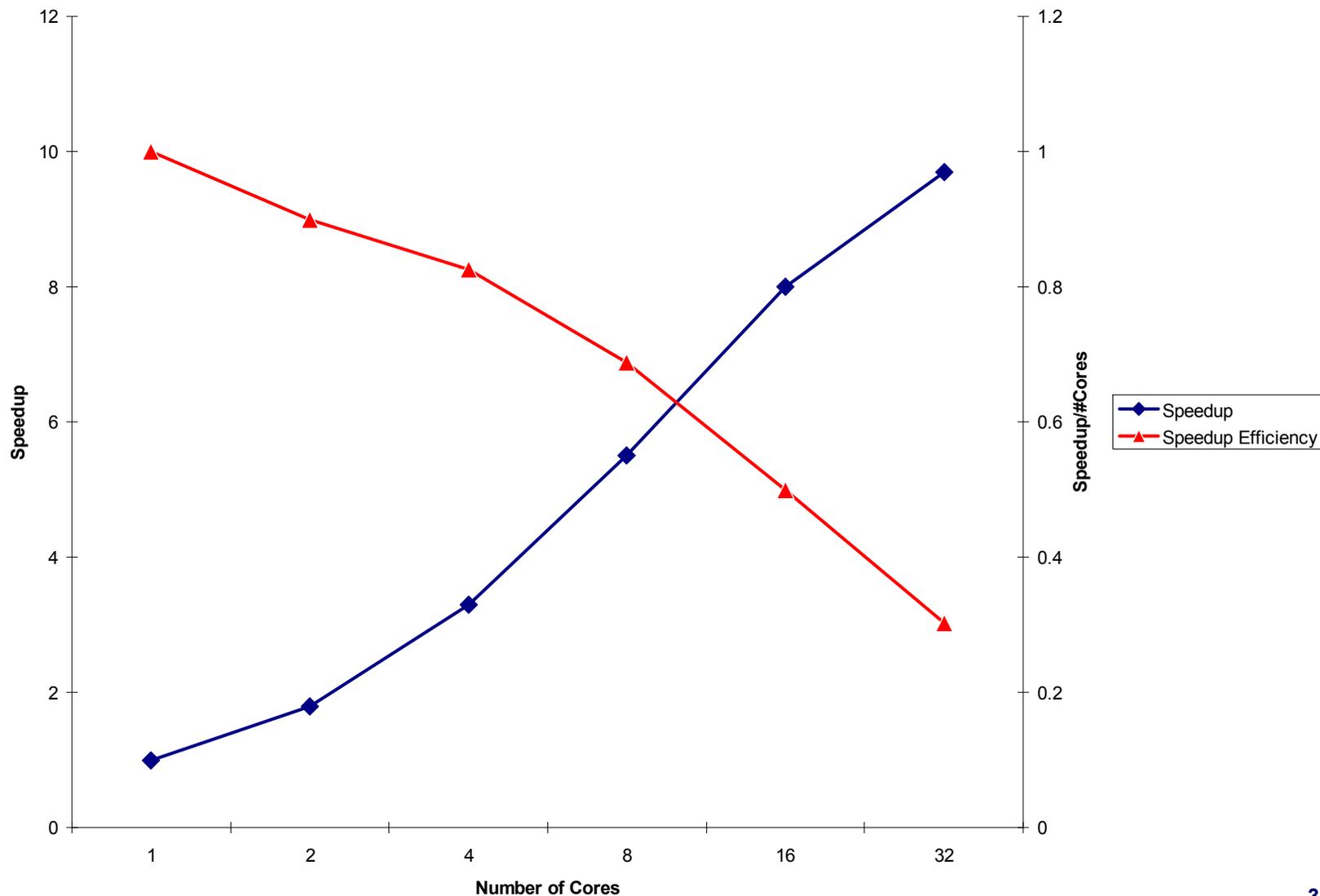


EO1 satellite



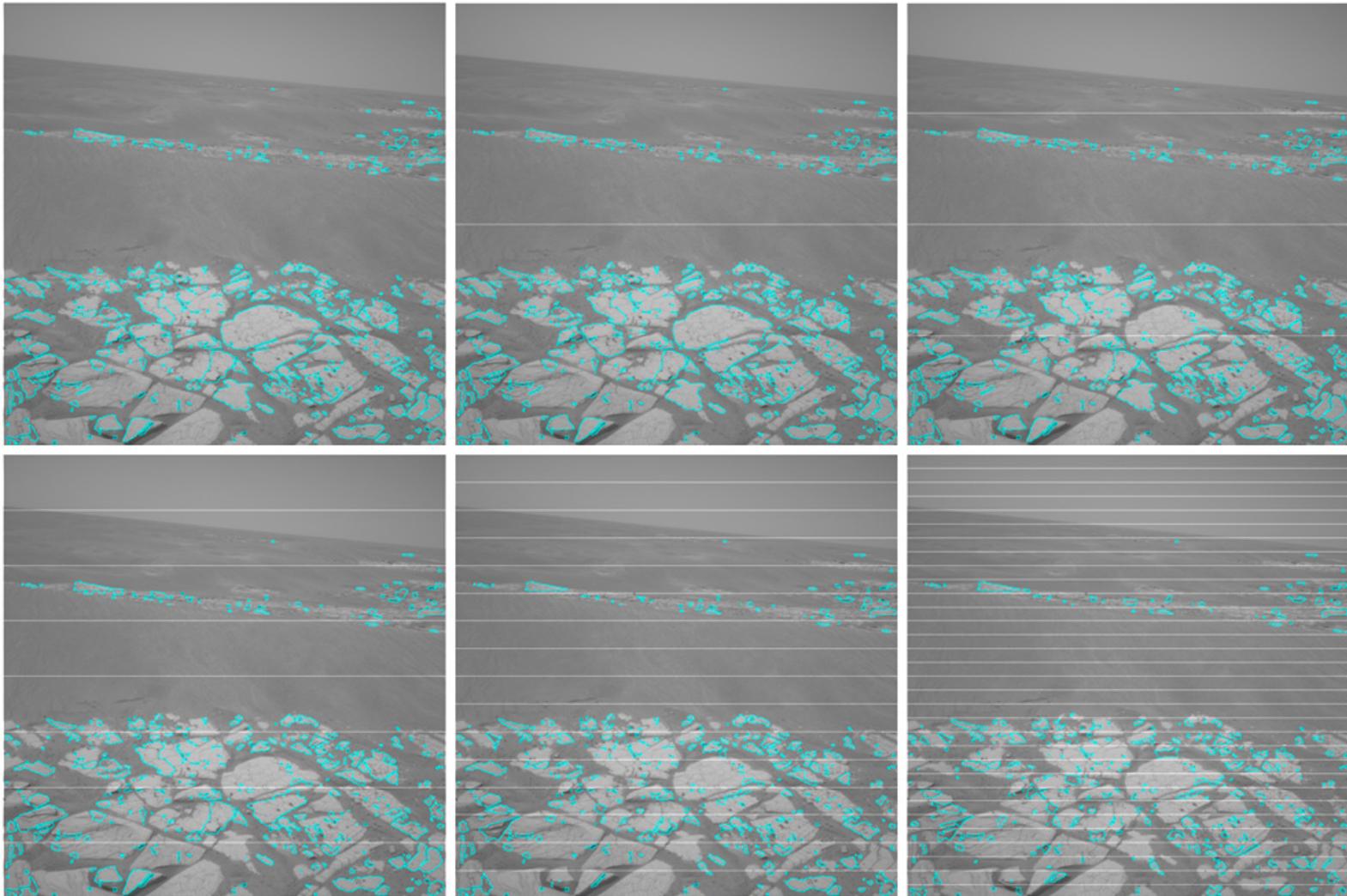
Scaling Results - Speedup

(Constant test set on differing # of cores)





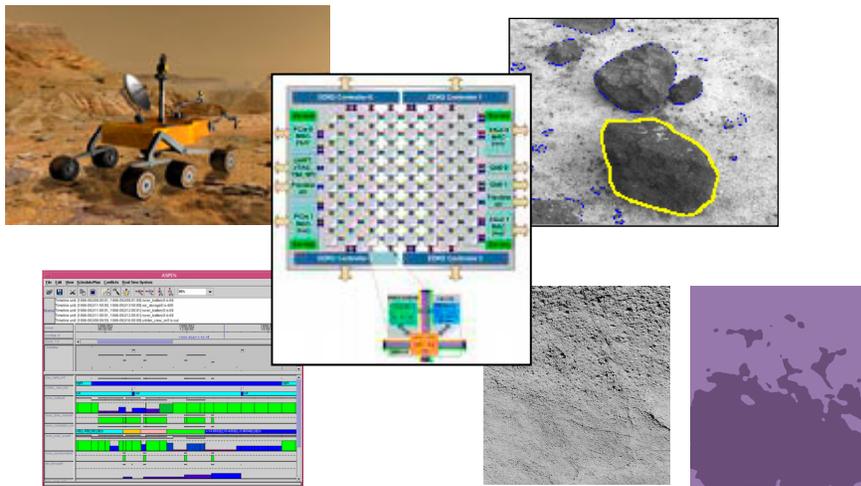
Sample Result



Improving Long-Range Rover Science Using Multi-Core Computing



Jet Propulsion Laboratory



Objectives:

- Develop and demonstrate key capabilities for rover traverse science using multi-core computing
- Adapt three autonomous science technologies to SOA multi-core system
- Demonstrate with rover hardware and measure performance benefits using metrics such as execution time and data processed

Task Manager:

Tara Estlin (818) 393-5375
Tara.Estlin@jpl.nasa.gov

Task Members:

Benjamin Bornstein, Brad Clement,
Paul Springer, Robert C. Anderson

Participating Organizations:

JPL (317, 388, 322)

Testbeds/Facilities:

Tilera Tile64, FIDO rover (JPL Mars Yard)

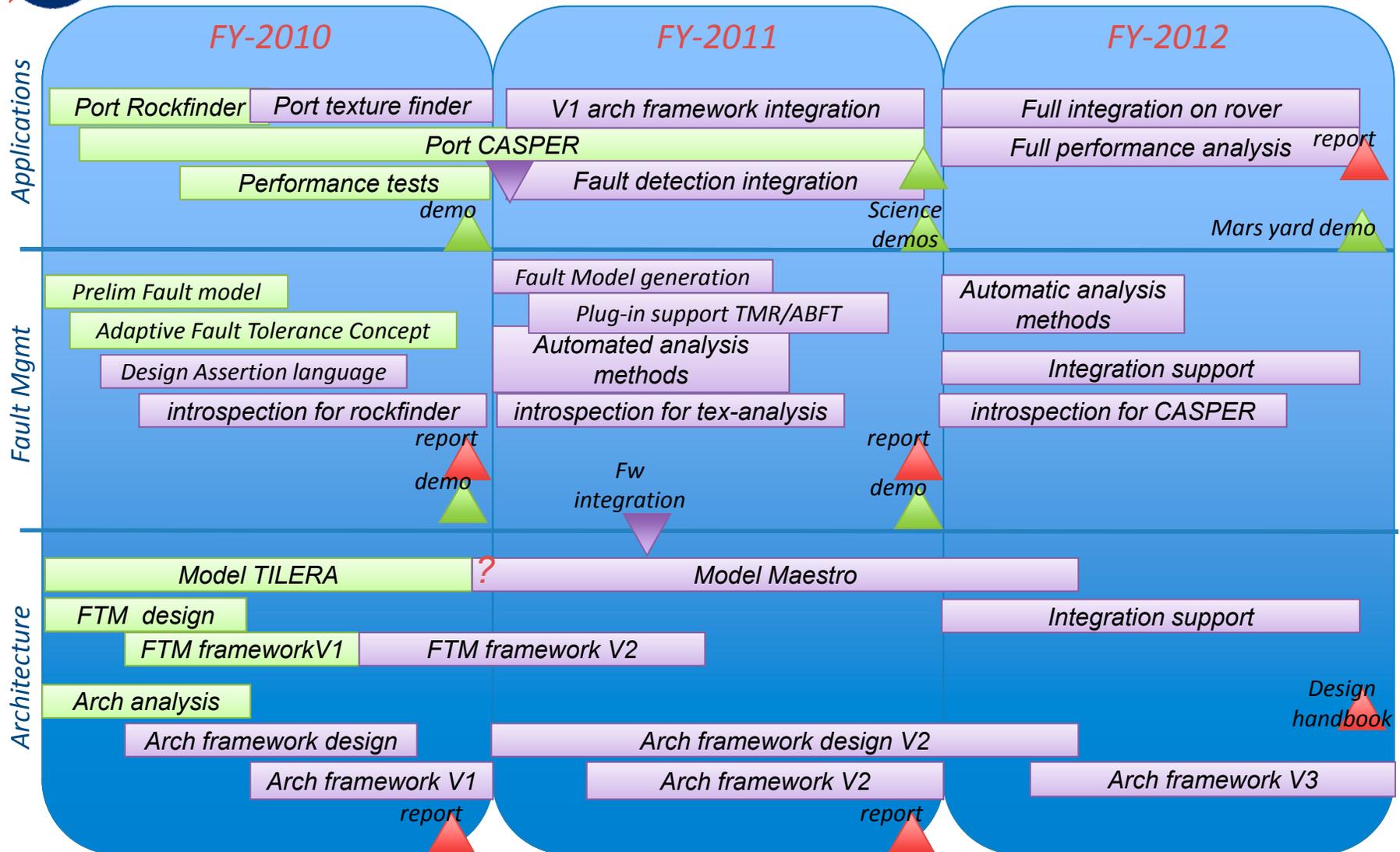
FY10 Funding (\$K): \$195K

Schedule/Key Milestones:

- Adapt one image analysis capability (for rock-finding) to multi-core processor
- Deliver set of architectural requirements and use cases to System Design task
- Adapt second image analysis capability (for texture analysis) to multi-core processor
- Demonstrate image analysis capabilities on multi-core processor using MER navigation camera images or using rover hardware



3-Year Task Plan



4/14/2010



FY10 Task Schedule



FY10 Milestones	Planned Date	Actual Date
Adapt image analysis capability for rock detection on multi-core processor and test in simulation	March 2010	Complete
Deliver set of architectural requirements to System Design task	March 2010	Complete
Perform initial analysis of CASPER planner and determine top strategies for multi-core	Bonus	Complete
Adapt image analysis capability for texture classification to multi-core processor and test in simulation	Sept 2010	
Demonstrate image analysis capabilities on multi-core processor using MER navigation camera images or rover hardware	Sept 2010	



Plan for Quarters 3 and 4

- Design and develop implementation of terrain texture classifier on multi-core processor
- Evaluate multi-core benefits for texture classifier
- Evaluate image tiling and optimal cache usage for rock detection parallelization
- Create design document for CASPER describing adaption options for multi-core
- Continue providing inputs to System Design task and Fault Tolerance task



FY11-12 Task Schedule

FY11-12 Milestones	Planned Date	Actual Date
Integrate image analysis capabilities with architecture using API from System Design task	March 2011	
Adapt automated planning system to multi-core processor and integrate w. multi-core software architecture	Sept 2011	
Demonstrate automated planning on multi-core processor using rover hardware. Measure performance improvements.	Sept 2011	
Fully integrate autonomy techniques on multi-core processor into one integrated system using API from System Design task	March 2012	
Perform rover hardware demonstration in JPL Mars Yard highlighting performance improvements for all autonomy capabilities fully integrated on multi-core processor. (Demonstration will be collaboration between all initiative tasks.)	Sept 2012	
Deliver report describing full performance benefits achieved through multi-core integrated system	Sept 2012	
Deliver report describing performance trade space given diff. multi-core configurations and power requirements	Sept 2012	