# Performance evaluation of ASCI benchmark AZTEC using Paradyn

**Final Report** 

Instructor: Dr. Frank Mueller (<u>mueller@cs.ncsu.edu</u>)

Team Members:

Raj kumar Nagarajan (<u>rknagara@cs.ncsu.edu</u>) Vikram S Poojary (<u>vspoojar@cs.ncsu.edu</u>)

#### **Problem Description:**

The main objective of this project is to use Paradyn, a parallel performance tool, to analyze and evaluate the performance of ASCI benchmark AZTEC. And to identify the bottlenecks in the benchmark application, to improve the performance by algorithmic changes or MPI specific changes, suggest improvements to the tool based on experience with benchmark evaluation.

In this document we discuss the following:

- AZTEC Benchmark
- Performance of AZTEC on the cluster
- Paradyn
- Evaluation of AZTEC using Paradyn
- Improvements to tool
- Location of src and binary files on cluster

### AZTEC:

AZTEC is a massively parallel iterative solver library for solving sparse linear systems. It provides state-of-the-art iterative methods that performs well on parallel computers (applications of over 200 Gflops have been achieved on the Sandia-Intel TFlop Computer) and at the same time is easy to use for application engineers.

Simplicity is attained using the notion of a global distributed matrix. The global distributed matrix allows a user to specify pieces (different rows for different processors) of his application matrix exactly as he would in the serial setting (i.e. using a global numbering scheme). Issues such as local numbering, ghost variables and messages are ignored by the user and are instead computed by an automated transformation function (AZ\_transform). Efficiency is achieved using standard distributed memory techniques; locally numbered submatrices, ghost variables, and message information computed by the transformation function are maintained by each processor so that local calculations and communication of data dependencies is fast. Additionally, AZTEC takes advantage of advanced partitioning techniques (Linear and Box partitioning) and utilizes efficient dense matrix algorithms when solving block sparse matrices.

### Performance of AZTEC on the Cluster:

The test program solves Poisson's Equation, using Finite Difference on an n \* n \* n Grid. The Grid Sparse matrices are represented using DMSR (Distributed Modified Sparse Row) format, which is a slightly modified format of MSR. In the test experiment, the cluster had 13 active processors running.



## Test with varying Grid size and with 128 PDE's and Linear partition of matrix:

Observation:

- 1. Performace increases with increase in the number of processors.
- 2. But the performance incease is small as the Grid size increases. This is because of the reduction in the parallel computation on the cluster. Each node has to compute a large Grid Size.
- 3. For number of processors >= 20, the performance comes down. This is because of the inherent limitation of the cluster size.

### Test with varying PDE's and with a GRID size of 216 and linear partition of matrix:



### Test with varying Grid size and with 128 PDE's and Box partition of matrix:

Grid Size	8 Proc
8	71.66
64	281.9
216	324.5
512	316.88

Observation:

1. The Box partition gives better performance than linear partition. But it is severly limited by the number of Grid size patterns and Processors that can be used. The Grid Size needs to be a cube-root of some integer (box size across each dimension) and number of processors should be an multiple of grid size.

### Paradyn:

It is a tool for measuring the performance of parallel distributed programs. It achieves this by dynamically inserting (attaching) the instrumentation code to unmodified executable. The instrumentation is done automatically by the Performance Consultant module, which identifies the performance problems, decides where and when to collect data. It uses a W3 (Why, Where and When) Search model. The tool also provides an open interface for program visualization and can be configured for application specific performance data.

### Evaluation of AZTEC using Paradyn:

The AZTEC benchmark program was run with the following parameters under Paradyn:

Num of processors: 5 Finite Difference MSR Poisson method Grid Size: 27000 Num of PDE: 256 Linear partition

For detailed instructions to run the benchmark under Paradyn, refer to the readme file.



The following are the screenshots of the program execution:

Fig 1:

The above screenshot depicts the process of launching the benchmark under paradyn.

4						5										
	whereAxis				_ <b>=</b> ×											
vs	Paradyn				1			= X								
4			Parai	lyn Main Cor	trol	¥\$,	P	ara								
	File S	etup	Phase	Visi		Help	dyi	r"								
	UIM status		: ready													
	Application 1	name	: program:	mpirun, ma	chine: os05, use	r: vspoojar	, daemon	mp								
	Bata Manager	scatus	: readu													
	Processes		: MPICH													
	152 14 53	70	• PTD=7260	readu												
	152.14.53	.75	: PID=8483	, readu.												
i.	152.14.53.	72	: PID=519:	, ready.												
E	7 152.14.53.	.74	: PID=1565	il, ready.												
	RUN	ſ	PAU	3E [	EXPORT		EXIT									
		1				-										
	Ctri-double-cli	CK TO SE	electrun-sei	ect all subtre	ees of a hode											
	Hold down	Alt and	d move the	mouse to sc	roll freely											
	Ctrl-Sing	le-Click	right butto	n to show V	isi table											
-			-		(1997)	emacs.	1 5	Gnome		ztec 1	IN THE	etsca 1		etsca 1		04-07 PM
2	1 3 2 1	21	10	(?)		[ennates.		anomo			and a line		and the	0.000j	╤(	Eri Mau 03
				-		🐨 (vspooja	a]	Paradyn	П [Те	erm Win]	wł 🔟	nereAxis				i ni way uz

*Fig 2: Application Initialized and ready to execute* 

The above screenshot depicts the stage when the paradynd daemons have been launched on all the nodes to monitor the MPI processes, the daemons have attached to the processes and before the steps for gathering performance metrics have been carried out.

whereAxis										
where Axis   Selections Navigate   Id-linux.so.2   libc.so.6   libc.xa.so.3   libgm.so.0   libm.so.6	Whole Program   Code   DEFAULT_MODULE   AZ_set_VBR   AZ_set_comm   AZ_set_precond_print_string   AZ_set_proc_config   AZ_set_proc_config   AZ_set_st_solver_parameters   AZ_setup_diclap_msr   AZ_solve_subdomain   AZ_sort_dible   AZ_sort_lints   AZ_sortqlists   AZ_space_for_factors   AZ_space_for_padded_matrix	Machine 152.14.53.70 152.14.53.72 152.14.53.74 152.14.53.75 152.14.53.75 152.14.53.75 Semaphore SpinLock								
	AZ_spitup_big_msg AZ_srandom1 AZ_subMSR_getrow									
	M 47 suhMSR matvae mult									
Search:										
	Click to select: double-click to exnand/un-exn	and								
Shift-double-click to expand/un-expand all subtres of a node										
Ctrl-double-click to select/un-select all subtrees of a node										
	Hold down Alt and move the mouse to scroll fr	eely								
	Ctrl-Single-Click right button to show Visi tab	le								
ي 🍕 🍋 🛄 🎝 🛄	Gnome Cal] 📀 [Aztec_pubs] 😻	[Netscape:] 🗱 [Netscape: ]	04:25 PM Fri May 02							

Fig 3: Components of the Program (Machine and Program functions)

The above screenshot details the where axis of the search space in W3 model of paradyn. In this screenshot, we have enabled node 152.14.53.70 and the function AZ\_solve as the specific foci for which we want to gather metrics.

shg	
The Performance C	onsultant
Searches	dvn <sup>ata</sup>
Current Search	n: Global Phase
Caligraph-based search for Global Phase	
	4
TopLeve	elHypothesis
ExcessiveSyncvvaltingTime	CPODound
	×
Resume	Pause
Never Evaluated	instrumented
Unknown	uninstrumented
True	instrumented; shadow node
False	uninstrumented; shadow node
Why Axis Refinement	Where Axis Refinement
Hold down Alt and move	the mouse to scroll freely
Click middle button on a n	ode to obtain more info on it

Fig 4:Initial Performance Consultant – before application run

The above screenshot is of launching the performance consultant window just before running the benchmark under paradyn. We can see that the initial hypotheses have not been tested by means of the green color on the hypothesis nodes.

Term Win	Application Output		Para		
Fi Paradyn					
Fal. Fal. Fal.	Paradyn Main Cont	rol	Paradyn Metrics Menu		
Fai File Se Fai File Se Fai UTM status	etup Phase Visi : ready	1		Select Metrics and Focus(es) be	low
Application n Application s	ame : program: mpirun, mac status :	hine: 0505, user: vsp	inumber_of_cpus	cpu_inclusive	_ cc_msgBytesRecv
al Data Manager	: ready • MPTCH		pause_time	_ sync_ops	_ msgs
1 152.14.53.	70 : PID=7260, ready.		active_processes	_ sync_wait	msg_bytes_sent
152.14.53.	75 : PID=8483, ready.		predicted_cost	sync_wait_inclusive	_ msg_bytes_recv
152.14.53. 152.14.53.	72 : PID=5191, ready. 74 : PID=15651, ready.		observed_cost	_ pp_msgs	_ io_ops
DIIN	DAUSE	EVPOPT	☐ func_calls_to	_ pp_msgBytesSent	🔄 io_wait
	THUGE		☐ func_calls_by	_ pp_msgBytesRecv	io_wait_inclusive
			exec_time	⊥ cc_msgs	🔄 io_bytes
			🗇 cpu	☐ cc_msgBytesSent	
			ACCEPT	CLEAR	CANCEL
<u>َ</u> کُنَّ 🖸	2 🌒 🦉 🖉		mac] 🧠 [Gno] ៚ [Azt spo] 🔲 Parad 🔲 Terr	e] 🐺 [Nets] 🐺 [Nets] n 🗖 Parad 🔲 [wher]	[shg] 04:1

Fig 5: Paradyn Metric options – To plot histogram

The above screenshot is one of enabling the metrics cpu\_inclusive and sync\_wait\_inclusive to be collected for the where axes listed in an earlier step.

After this step, we can run the benchmark under paradyn.

4		2		
0.000	whe	ereAxis		
vs	Sele	ctions Navigate		
	$\overline{\Delta}$	shg	_ = ×	
Î		The Performance C	onsultant	
		Searches	d ara	
		Current Search		
		Callgraph-based search for Global Phase		
	5 - F			
	78			
ŝ	Searc		)A	
			alHypothesis	
		ExcessiveS	vncWaiting Time	
		Excessivel	O Blocking Time	
		CPUbound		
	_			
		Resume	Pause	
		Never Evaluated	instrumented	
		Unknown	uninstrumented	
		False	uninstrumented; shedow hode	
		Why Axis Befinement	Where Axis Refinement	
		Hold down Alt and move	the mouse to scroll freely	
		Click middle button on a n	ode to obtain more info on it	
	ļ,			
0				
0	] 4	j 👂 🍕 🧉 🛄	🛛 🖾 [Gnome] 🧼 [Aztec] 🐺 [Netsca shg 🥥 [vspooja] 🗖 Paradyr	] 🐺 [Netsca] 🔲 [Histogr]
		Fig 6: Performa	ance Consulatant – A	fter Program Run

The above screenshot shows the conclusion of the Performance consultant when the application has exited. The above screenshot shows that all of the top-level hypotheses have evaluated to false, implying that the performance consultant is unable to find any bottlenecks for the benchmark.

	vspoojar@os	05:~/aztec	/app								X								
	Histogram											×							
			1	Histogran	ı Visu	alization					Pa	ra							
	File Curv	re									lyñ	<u>i</u> a							
	CPUs																		
	1.1																		
	1-				ſ		-												
	0.9																		
	0.7 -																		
	0.6 -																		
	0.5 -										Z								
	0.4 -										0								
	0.3 -																		
p	0.2 -																		
	0.1 -																		
	0+	0:40	1-20	2:00	2-40	3-20	4-00	4:40	5-20	6-00	-								
		0.10	1.20	2.00	2.10	Min: Sec	1.00		5.20	0.00									
	срц_і	nclusive< (Code)	DEFAULT_MO	DDULE/A Z_solv	e./Machine	/152.14.53.70.>	(smoothed)												
	sync	wait_inclusives	CODEDEFAU	ULL_MODULE/A	_solve./m	achine/152.14.5	3.70.> (smoo	)med)											
					P	'an													
	Dhace Global																		
		daliyon	ad		_														
	0 total p	oints us	ed i																
	54 mini-t	ramps us	ed																
	3010 tran 7 ptpace	ip bytes other ca	116																
	3629 ptra	ace write	calls																
	23312 ptr	ace byte	s writte	en															
	0 1nstruc	time use	herated d to ge	nerste in	etrume	ntation													
	0.000400	0100 000	501	101 0 00 211	o or ano	1100101011					⊡								
-																			
100		-		-			8		<b>1111111111111</b>		k		. Not	P.L.L.	. (5	1.12-1			
		2.			2	1		inome]	V [Azt	ec] 👔	۶۳ [N	etsca.	]	[Netsca.		Histogram	-(		04:27 PM
1		-	0					[shg]	🛃 vsp	oojar 🛙	🗌 [P:	arady	n] 🗖	Term W	in 🔲	[whereA		-	militay 02

Fig 6: Histogram Plot of CPU time and Sync Wait time

The above screenshot shows the behavior of the benchmark on one of the cluster nodes for the function AZ\_solve. This shows that the function is CPU bound as indicated by the blue line and there is no synchronization overhead as indicated by the yellow line.



The above figure shows that Paradyn outputs the overhead due to its instrumentation when the benchmark has finished executing.

So, in conclusion, the performance consultant did not show up any bottlenecks for the benchmark. Consequently, we did not have a starting point to start a detailed manual search by generating Visi graphs for each of the points on the W3 search space. We did however carry out an evaluation for the function AZ\_solve by plotting the graphs that is shown in one of the screen shots above.

#### Suggestions for functionality Improvements to Paradyn:

- Multithreading support is currently only available for Solaris platforms. It needs to be enabled for all the platforms and new abstractions need to be added to the W3 search model to account for multithreading.
- It should be possible to restart applications from Paradyn itself. Currently, it does not allow us to restart the application even the application has finished executing under the tool.

### Location of Files in Cluster:

Aztec files: /home/vspoojar/aztec Test files: /home/vspoojar/aztec/app Library files: /home/vspoojar/aztec/lib

Paradyn files: /home/vspoojar/paradyn

Binary files: /home/vspoojar/paradyn/bin/i386-unknown-linux2.2

Source files: /home/vspoojar/paradyn/src/core Test files: /home/vspoojar/paradyn/apps

Readme file for this project: /home/vspoojar/paradyn/README