Parallel Space-Time Adaptive Processing On a Cluster of Personal Computers

Fernando Silva, Gary B. Lamont

Air Force Institute of Technology
Wright-Patterson AFB OH
{fsilva, gary.lamont@afit.af.mil}

Abstract—

This work evaluates the capabilities and the performance of the Air Force Institute of Technology's Pile-of-PCs for parallel digital signal processing using space-time adaptive processing (STAP) under the Linux OS. The MITRE RT_STAP Benchmark version 1.1 is ported and executed on it, as well as on a cluster of six Sun SPARC workstations connected by a Myrinet network (the AFIT NOW), and on a IBM SP for comparison. Modifications to the RT_STAP benchmark source code are performed to accommodate the BLAS routines obtained from the US Department of Energy's Accelerated Strategic Computing Initiative, and the FFTPACK from the Netlib repository, allowing improvements in the sustained Gflops/sec rates. However, the Pile-of-PCs also reveals limited scalability as a result of severe communication overheads imposed by RT_STAP cornerturn operations. Analysis of experimental data indicates that the PC Cluster outperforms AFIT NOW but needs interconnection network improvements to be globally competitive to multicomputers such as the IBM SP.

Keywords— **Cluster of PCs, parallel signal processing, STAP, real-time benchmarking, Linux.**

I. Introduction

Space-time adaptive processing (STAP) is a well-known stochastic signal processing technique in the area of airborne surveillance radars, which is used to detect weak target signal returns embedded in strong ground clutter, jamming, and receiver noise. A significant feature of STAP is that it can improve the performance of airborne Electronically-Steered Array (ESA) antennas while requiring little or no modification to the basic radar design [WAR 94]. This technique takes advantage of both the spatial and Doppler diversity of target signal returns, clutter, and interference to extract the desired signal by adaptively combining samples of multiple radar channels and pulses to null clutter returns and interference. Processing data from multiple channels provides the radar an opportunity to control the spatial response of the system while processing multiple pulses enables the processing to separate signals based upon their Doppler frequencies. However, STAP consumes great amounts of computational resources, since an extremely large amount of data needs to be processed in real-time. This in turn requires a large computational throughput.

The experiences described in this paper represent the *first* attempt to address the computational capabilities and the cost/performance ratio provided by a cluster of personal computers when applied to STAP. It is also a step further in the direction of employing cluster of PCs instead of massively parallel processors or digital signal processors in real-time environments. Although difficulties exist – such as operating system overhead and implementation efficiency – the fast development time, flexible nature of software, and increased speed and affordability of general-purpose microprocessors make the use of these platforms desirable.

The rest of the paper is organized as follows: in Section II, we present the features of the RT_STAP benchmark, and the computational platforms used in the research. Section III details the process of porting the benchmark into the PC Cluster, the nature of the modifications made to the benchmark's source code, and the experimental framework. Section IV reports our results and analysis, and Section V presents our conclusions.

II. Background

This Section is organized around two main parts. First, we discuss the critical aspects associated with the RT_STAP Benchmark. We finish the Section by describing the computational platforms to be used for benchmarking.

A. The RT_STAP Benchmark

The STAP implementation chosen is the Real-Time Space-Time Adaptive Processing Benchmark – RT_STAP – created at MITRE Corporation in Bedford, MA, and currently in version 1.1. It is a realistic compact application benchmark based upon data collected by the Air Force Research Laboratory's MultiChannel Radar Measurement Program (MCARM) airborne system. The RT_STAP version 1.1 is written in C, and provides sequential versions of algorithms implementing Displaced Phased Center Antenna (DPCA) processing, first-order post-Doppler, and high-order post-Doppler factored STAP, as well as concurrent versions of the last two algorithms mentioned.

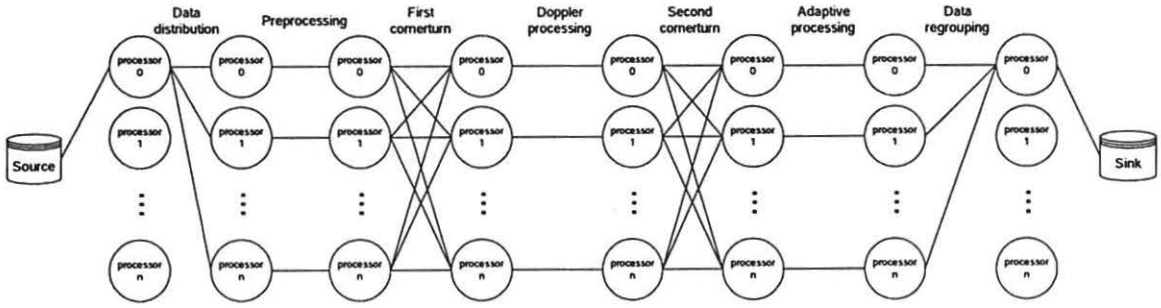


Fig.1. RT_STAP data-parallel programming model.

using MPI as the interprocess communication mechanism. The primary reason for the selection of RT_STAP is the fact that it was originally developed to evaluate high-performance computers (MPPs and Shared-Memory Multiprocessors) for STAP [HWA 96][MIT 99][WAN 97].

The modules that build the benchmark are: *preprocessing, nonadaptive Doppler filtering, adaptive processing,* and the global communication steps represented by the *distributed matrix cornerturns* (see Figure 1). The global exchange / reorganization phase (cornerturns) is accomplished through the use of messages, employing the *MPI-Alltoallv* collective operation.

The benchmark comprises three different STAP implementations, involving three levels of complexity: easy, medium, and hard. The easy benchmark corresponds to the post-Doppler adaptive Displaced Phased Center Antenna (DCPA) algorithm and requires a real-time computational throughput of 0.60 Gflops/sec. This case represents technology used in current radar systems. The medium benchmark case corresponds to the first-order Doppler-factored STAP and requires a throughput of 6.46 Gflops/sec. The hard benchmark case corresponds to an implementation of the third-order Doppler-factored STAP and requires a throughput of 39.81 Gflops/sec. The RT-STAP also includes the implementation of the data preprocessing typically performed before the application of nonadaptive filtering and subsequent adaptive processing.

For the first-order Doppler factored STAP and third-order Doppler factored STAP parallel implementations are available that support up to 64 processors and include software interfaces implementing function name resolution to allow the use of specialized linear algebra routines designed for the Sun©, Mercury©, and Sky© computing platform vendors. For the evaluation of the hard, medium, and easy benchmark cases, 22, 16, and 2 of the 22 available MCARM data collection channels were used, respectively. For all three cases, the CPI consisted of 64 contiguous pulses. The high performance computer must input 0.49, 3.93, and 5.41 Mbytes of real data per CPI for the easy, medium, and hard benchmark cases, respectively.

For RT_STAP both the period and latency are closely associated with the CPI of the radar system. The period corresponds to a single CPI, and according to the MCARM specifications, it equals 32.25 milliseconds corresponding to a CPI with 64 pulses. The latency case requirements dictates that data input, processing and writing to data sink must occur within 5 CPIs. This corresponds to a latency of $32.25 \times 5 = 161.25$ milliseconds for the MCARM. The operation rates are specified in billions of floating-point operations per second (Gflop/s) and are computed by dividing the floating point operation counts from the benchmark specifications in [CAI 97] by the period. For this scenario, the period is equivalent to the duration of the CPI and is 32.25 milliseconds.

B. Computational Platforms

In this subsection we objectively describe the configuration of the computational platforms used in this research [SIL 99]: the PC Cluster, the AFIT NOW, and the IBM SP. The AFIT cluster of PCs[†] is a continuously-evolving dedicated shared-nothing parallel machine consisting of one Dell 450 MHz Pentium II processor, six Dell 400 MHz Pentium II processors, one Dell 200 MHz Pentium processor, and four Gateway 333 MHz Pentium II processors connected via a 100 Mb/sec full duplex 24-port switched Fast Ethernet – the average delay through the switch is 11 microseconds. The switch has an aggregate internal bandwidth of 6.3 Gbit/sec and an aggregate network bandwidth of 800 Mbit/sec. Each processor can be booted either running Windows NT 4.0 or Linux 2.0.33 operating systems. Parallel communication is handled through MPICH version 1.1 for Linux applications. Three of the four Gateways have 128 Mb 15 nsec SDRAM, and each of the Dell processors has 128 MB of 10 nsec SDRAM. The fourth Gateway has 256 Mb 15 nsec SDRAM. The Pentium 200 MHz has 32 Mb of main memory.

The AFIT NOW consists of five Sun Ultra Sparc© workstations model 170 (170 MHz processor) and one of model 200 (200 MHz processor), connected via the high-speed Myrinet© switch. The processors are four-way superscalar of version 9, with two integer ALU units and

[†] Also named AFIT Bimodal Cluster (ABC) in this paper.

two pipelined FP ALUs. There is a 16 Kbyte direct-mapped data cache and a 16 Kbyte 2-way set associative instruction cache, both on-chip. The level-2 cache has 512 Kbytes. Each workstation has 128 Mbytes of RAM and two 1Gbyte local hard disk drive. The Myrinet network includes an 8-by-8 crossbar switch, and each link provides 1.28 Gbits/sec in each direction. The protocol used in the messaging layer is TCP/IP. The MPI implementation used for communication is MPICH 1.0.1.

The IBM SP system in the Aeronautical Systems Center Major Shared Resources Center (ASC MSRC), at Wright-Patterson AFB, is a scalable distributed memory multicomputer based on the IBM RS/6000 Power2 SC 4-issue superscalar processor operating at 135 MHz, and capable of deliver 540 Mflops peak. From its 256 processors, 233 are available for computation, each one with 1 Gbyte of main memory. The NIC includes a Power PC 601 processor that performs DMA. The interconnection network is a multistage Omega network, with theoretical bandwidth of 40 MB/sec per link in each direction. The operating system is the AIX 4.1.

III. Methodology

Investigating the use of the PC Cluster for STAP involves two basic steps: port the RT_STAP benchmark into the cluster and check for effectiveness and performance results. This last step is explored further, by modifying the source code of the original implementation in order to allow it to obtain improved performance from the platform to which it was ported, while maintaining the portability. Additionally, the results obtained from the *original* implementation of RT_STAP on the AFIT NOW and IBM SP provide additional insight on the characteristics of the benchmark and on the capabilities and scalability of the Pile-of-PCs. The software building process is directed to the use of the single-precision standard ANSI C implementation in order to reflect the default validation criteria of the benchmark, and to cope with the fact that the PC Cluster does not have a set of customized library routines to perform linear algebra and signal process computations at this stage of the process. The RT_STAP implementation supports self-validation as one of the option flags on the command line for execution, and this feature is used throughout to test for correctness of results.

C. RT_STAP Source Code Modifications

The movement of data between memory and registers can be as costly as arithmetic operations on the data. This cost provides considerable motivation to restructure the existing standard C implementation in order to benefit from the surface-to-volume effect. Close examination of the code reveals the existence of numerous vector-scalar, vector-vector, and matrix-vector routines; also, the fast Fourier transform (FFT) kernel is frequently used. Therefore, we provide the PC Cluster with a Basic Linear Algebra Subroutines (BLAS) Library and a set of customized FFT routines that could explore the capabilities of the Pentium II processor, and modify RT_STAP to make use of it, where applicable. We download and install a BLAS implementation from the Sandia National Laboratories [SAN 99], in Albuquerque, New Mexico. This Fortran implementation, in its version 1.1L and called *ASCI Red Pentium Pro BLAS*, is one of the libraries developed for the Intel ASCI Option Red Supercomputer, and is targeted to Unix-like environments. We also downloaded the necessary routines from a publicly available FFT package from the Netlib repository [NET 99] named *FFTPACK*. It is a widely used implementation based upon a radix-2 version of the algorithm, but not as efficient as the mixed-radix version [LAM 97].

D. Experimental Framework

For real-time signal processing applications, the goal of parallel processing is also to meet specified latency requirements. Therefore, the measure of the system scalability must take this factor into consideration by adopting a time constrained scaling approach that can alleviate the sequential bottleneck and improve speedup by scaling the problem size with the increase in machine size. The timing specifications for the RT_STAP benchmark emphasize a similar approach, by determining the smallest machine size that is required to meet a prescribed real-time constraint by using (scaling) different problem sizes, algorithm complexities, and latency constraints. Also, it is important to determine the overhead contributed by communication and I/O operations as a function of machine size.

IV. Results & Analysis: ABC

A naming convention to be adopted from now on is to call the first-order post-Doppler STAP simply by *FOPD*, and the high-order post-Doppler STAP as *HOPD*. We start by reporting the execution times obtained for the sequential versions of the DPCA, FOPD, and HOPD implementations on Table I. According to the execution times from the *original* (orig.) C implementations, these results represent improvements in sequential performance up to 16% for the *modified* versions (mod.).

In regard to the parallel benchmark programs, we first discuss the FOPD. The corresponding performance according to the benchmark requirements is listed on Table II, obtained for FOPD running on 7 processors - 06 Pentium 400 MHz and 1 Pentium 450 MHz. We used the faster processor only when running with seven machines, and we believe that the difference in performance is not significant to discard the assumption of a homogeneous environment. The percentages sustained show that the FFT

implementation was capable of meeting the requirements for Doppler processing.

A Gantt chart was built to describe how the execution times are spent among the different stages of the implementation, and the length of each bar represents the total elapsed time in seconds for each version and number of processors, as seen in Figure 3[†]. We can see that the program scales well up to 4 processors, when execution times start to increase because the latency of inter-process communication outperforms the reduction in computation times, affecting the program scalability.

The high-order post-Doppler corresponds to the hardest case between the benchmarks. It is a generalization of the algorithmic concept applied to FOPD, being more computationally intensive. A better computation / communication ratio for HOPD also allowed the Pile-of-PCs to show better scalability. This time, the program scaled up to 6 processors, mainly because higher computation rates tend to provide better scalability in network computers with high latency in communication like the PC Cluster.

In order to observe how the elapsed times are partitioned among the different stages of HOPD, as well as time spent in communication, we also built a Gantt chart for this STAP implementation. The graph can be seen in Figure 2. From the benchmark specifications, we see on Table III that modified HOPD was capable of meeting the real-time requirements (*% sustained*) only for the weights application stage, although it fell short for Doppler processing. A considerably larger number of processors is needed to meet the flops/sec rate for the weight computation stage, and this fact demands a machine with *different* and *improved* hard./soft. communication structure in order to allow the adding of more processors without compromising the gains in computation times. Also, a high level of uniprocessor performance executing QR decomposition is a decisive factor.

Considering the theoretical peak Mflops rate for the Pentium II 400 MHz is 400 Mflops (meaning an operation completed at every clock cycle), the maximum utilization rates achieved for FOPD and HOPD were 28% and 32% of the theoretical maximum, respectively. Although the interpretation of this metric depends a lot on the machine and the application, it is a good indicator of software performance tuning.

E. Other Platforms: AFIT NOW and IBM SP.

The main purpose of these experiments was to observe the effect that different interprocess communication latencies and processor capabilities could have on both application and machine scalability. Because STAP is a much more computational intensive application, the execution times obtained from these two platforms were greater than those obtained by using the Pile-of-PCs for the same number of processors (the workstations use 170 MHz Sparc processors, and the IBM SP uses 135 MHz processors), although the scalability results were different, and generally better. We start with the AFIT NOW. Execution times from original sequential STAP programs run on average 2.8 times faster on the PC Cluster. Table IV shows descriptive statistics for parallel RT_STAP on the NOW. Differently from The PC Cluster, the FOPD program had its execution times reduced when more than 4 processors were used (see Figure 4). The reasons for these differences reside in better communication scalability and lower overhead provided by the pair Myrinet-TCP/IP. Although here the protocol is again a bottleneck that does not allow realization of better communication rates, we were able to get good results, specially for more than 3 processors. This occurred because the times spent in cornerturn operations and source/sink communication experienced improvements from 4 to 6 processors, as described in Figures 5 and 6. Specifically for more than 4 processors, source/sink communication takes less time on the NOW.

The other platform used for comparison was the IBM SP. The numbers show that the Pile-of-PCs is on average 2.4 times faster than IBM SP when running the original sequential implementations. On the other hand, the IBM SP demonstrated much more scalability in its interconnection network. We executed the parallel implementations of FOPD and HOPD using up to 64 processors (the limit imposed by the implementation). Figure 7 shows the charts for execution times obtained on the IBM SP. The same data for the PC Cluster is shown for comparison. The FOPD implementation running on the IBM SP could not meet the performance obtained by the PC Cluster running with four processors, and the reason for that was I/O. The IBM SP spent more time in reading the input datacube, the parameters file, the filter coefficients, and the steering vectors. The sum of the time spent on these I/O tasks were on average 8.5 times higher than on the Pile (this average considers HOPD I/O times as well), and the effect of this higher latency was worse on FOPD because I/O ended up encompassing a larger part of its overall execution time, as the number of processors increased.

In theoretical parallel computing, a common belief is that communication overhead increases with increasing machine sizes, but that was not totally true for FOPD and HOPD. As the scatterplot on Figure 8 shows, the average cornerturn times actually *decrease* as the number of processors increase; this is attributed to the decreasing message size. Other observation that can be made is that the

[†] The time that is accounted as miscellaneous is that relative to memory allocation/free time, generation of coefficients, and time spent to check the validity of input parameters. The item disk I/O encompasses time spent in reading the parameters file, the input data cube, the filter coefficients, and steering vectors.

performance of collective operations on the Cluster of PCs degrades fastly with the increasing number of processors involved due to network contention. When source/sink communication is considered, the theory completely reflects practice: the more processors are added, higher is the time needed to enable communication between all of them and the root processor. However, the effects of this endpoint contention on elapsed times is much less intensive on the IBM SP when compared to Pile-of-PCs, as sketched on the scatterplot in Figure 9.

The results obtained for 64 processors, according to the real-time requirements of the benchmark, are described on Table V. From that Table, the maximum sustained rate was 6 Gflops/sec during the weight computation stage of HOPD. Dividing this value evenly by the 64 processors results in 94.2 Mflops performed by each processor, which translates in 17.4% utilization for the POWER2 SC processor. The result obtained for the weight computation phase on HOPD (18%) shows that we are still far from meeting the throughput requirements in terms of QR decomposition operations, and that the machine can still scale up to the hundreds, relying on a sufficiently large problem size.

V. Conclusions

A simple cost/performance analysis between the Pile-of-PCs and the AFIT NOW clearly indicates the first as the winner. We can derive a value around $30/Mflop/sec for the PC Cluster running RT_STAP: a maximum sustained rate of 902 Mflop/sec in the weight computation phase dividing $27,700 – the total cost for hardware and software. The same evaluation done for the AFIT NOW produces a much higher ratio of approximately $650/Mflop/sec.

The Pile-of-PCs and the Linux OS provided a stable and flexible environment for development and testing. However, the MPICH implementation running upon the TCP/IP protocol could not utilize the full bandwidth that can be delivered by the Fast Ethernet interconnection. Identical experiments done on the AFIT NOW showed that the TCP/IP was a bottleneck in this process. The interconnection network imposed a severe negative impact on the scalability of the Pile, and this process seemed to be accelerated by the fast speed of the Pentium CPU as the machine scaled up, specially for relatively less computationally intensive applications like first-order post-Doppler STAP.

Collective communication and reduction operations were significantly affected, and showed rapid degradation as the machine size increased. Comparisons made using the results from the IBM SP showed that the reduction in message size did not bring benefits for the cornerturn operations on the Cluster of PCs. That indicates the current network latency need improvements to allow the cluster to benefit from the reduced size messages as the system scales up. We also experienced some level of network contention during collective communication operations on the Pile. Endpoint contention is another limitation present on the PCs' interconnection network. In this case, a change on the topology (such as to a fat tree) may provide relief to this problem by allowing an efficient implementation of software combining trees.

The positive results obtained by using the ASCI Red Pentium Pro BLAS and the FFTPACK packages are examples of effective software technology tracking that can enhance program performance without sacrificing portability. As the use of COTS hardware/software becomes mainstream, demonstrating easy-to-develop portable software for parallel computers is more important than creating complex optimized particular solutions.

Acknowledgments

We gratefully acknowledge Dr. Richard Linderman and Mr. Zen Pryk, from AFRL Rome Laboratories, for their general support, and for providing the RT_STAP Benchmark and associated input data set. Capt. Fernando Silva was sponsored by the Brazilian Command of Aeronautics, under the ITO no. BRTBRDF0125.

References

[CAI 97] CAIN, K. et al. RT_STAP: Real-Time Space-Time Adaptive Processing Benchmark. *Technical Report MTR 96B0000021*, MITRE Corporation, Bedford, Mass., Feb. 1997.

[HWA 96] HWANG, K. et. al. Benchmark Evaluation of the IBM SP2 for Parallel Signal Processing. *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 5, pp. 522-536, 1996.

[LAM 97] LAMONT, Gary. B.; GALLAGHER, David, M. Scalable Distributed Multi-Dimensional FFT Algorithm Design, Implementation, and Analysis. *Interim Report 2*, Air Force Institute of Technology, Summer/Fall 1997.

[MIT 99] The MITRE Corp. Benchmark Results, available at *http://www.mitre.org/research/hpc*, 1999.

[NET 99] NETLIB Repository at UTK, available at *http://www.netlib.org*, 1999.

[SAN 99] SANDIA National Laboratories, available at *http://www.sandia.gov*, 1999.

[SIL 99] SILVA, Fernando. Parallel Digital Signal Processing on a Network of Personal Computers – Case Study: Space-Time Adaptive Processing. *MSCS Thesis, AFIT/GCS/ENG/99J-01*, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, June 1999.

[WAN 97] WANG, C. J. et. al. STAP Benchmark Evaluation of the T3D, SP2, and Paragon. *Proceedings of the 10th International Conference on Parallel and Distributed Systems*, New Orleans, Oct. 1997.

[WAR 94] WARD, J. Space-Time Adaptive Processing for Airborne Radar. *Technical Report 1015, MIT Lincoln Laboratory*, DTIC n. AD-293032, 1994.

TABLE I

RT_STAP SEQUENTIAL PERFORMANCE: ABC

Prog.	Avg. exec. time	Std. Dev.	95% CI (+/-)
DPCA	0.3530	0.0017	0.0006
FOPD	2.4610	0.0007	0.0003
HOPD	10.7720	0.0026	0.0009

TABLE II

FOPD PARALLEL PERFORMANCE: ABC (7 PROCESSORS)

first-order post-Doppler	Flops Count	Exec. Time (sec)	Benchmark Requirement (Gflops/sec)	% sustained
Video to I/Q conversion	57,016,320	0.157	1.77	20.52
Array calibration and Pulse comp.	67,633,152	0.086	2.10	37.45
Doppler processing	15,728,640	0.030	0.49	100.00
Weights computation	63,700,992	0.062	1.98	51.89
Weights application	3,932,160	0.005	0.12	100.00

TABLE III

HOPD PARALLEL PERFORMANCE: ABC (7 PROCESSORS)

high-order post-Doppler	Flops Count	Exec. Time (sec)	Benchmark Requirement (Gflops/sec)	% sustained
Video to I/Q conversion	78,397,440	0.208	2.43	15.51
Array calibration and Pulse comp.	92,995,584	0.115	2.88	28.08
Doppler processing	21,626,880	0.040	0.67	80.70
Weights computation	1,074,991,104	1.192	33.33	2.71
Weights application	16,220,160	0.022	0.30	100.00

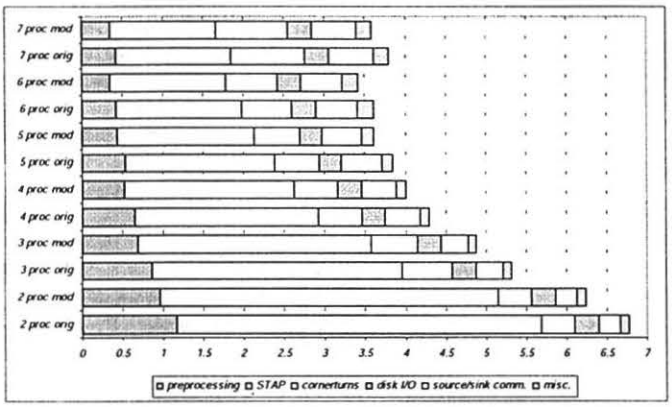


Fig. 2. Partitioning of the elapsed times for HOPD on ABC.

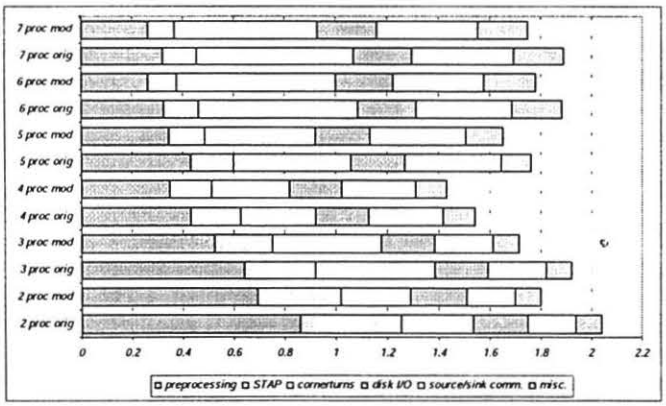


Fig.3. Partitioning of the elapsed times for FOPD on ABC.

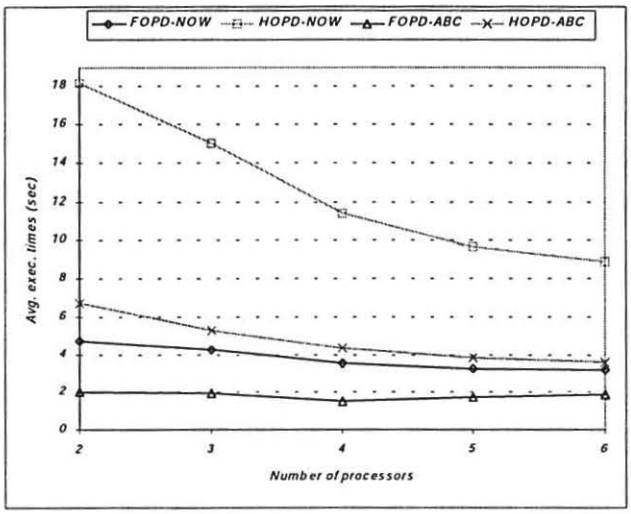
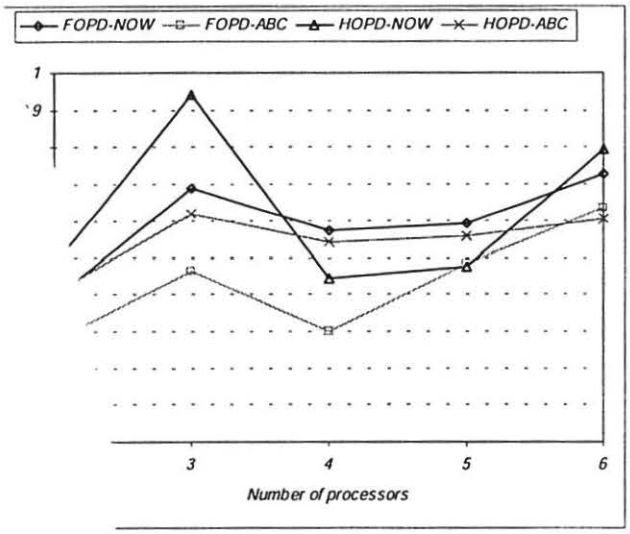


Fig.4. Comparative execution times: NOW vs. ABC.



spent in cornerturn operations: NOW vs. ABC.

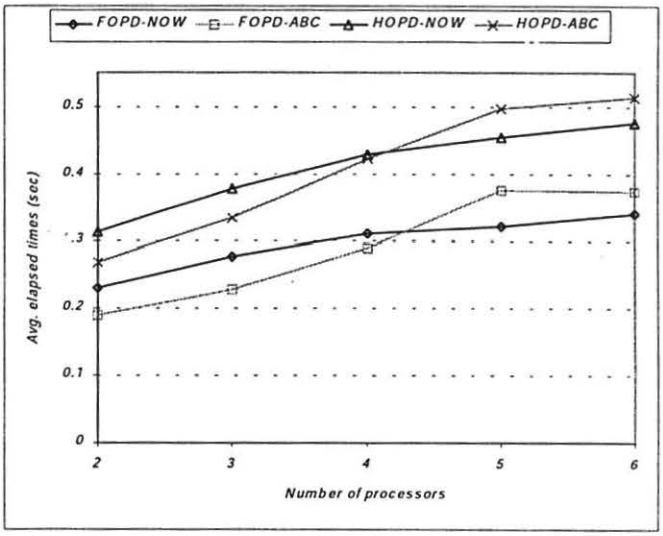


Fig. 6. Time spent in source/sink communications: NOW vs. ABC.

TABLE IV

DESCRIPTIVE STATISTICS FOR PARALLEL RT_STAP ON THE NOW

Program	number of processors	2	3	4	5	6
FOPD	avg. execution time	4.672	4.216	3.556	3.240	3.217
	Standard deviation	0.123	0.152	0.093	0.079	0.094
	95% confidence interval (+/-)	0.139	0.172	0.105	0.090	0.107
HOPD	avg. execution time	18.184	15.005	11.401	9.677	8.942
	standard deviation	0.291	0.112	0.220	0.147	0.126
	95% confidence interval (+/-)	0.329	0.127	0.249	0.166	0.142

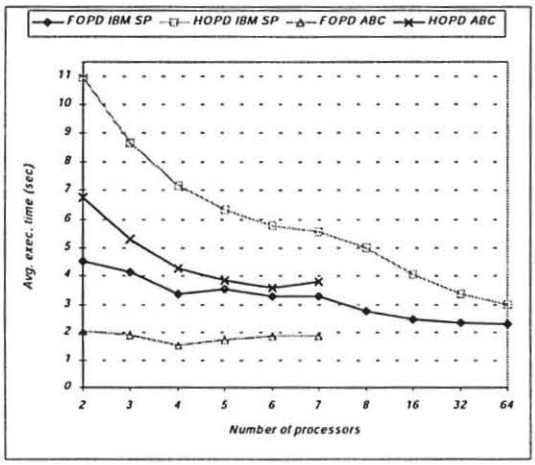


Fig. 7. Comparative execution times: IBM SP vs. ABC.

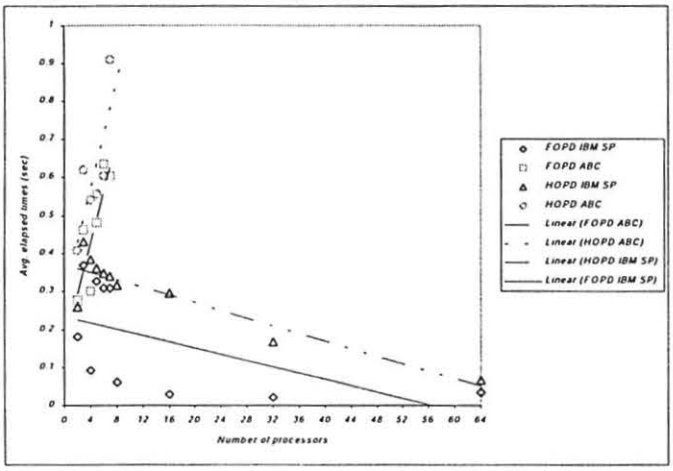


Fig. 8. Time spent in cornerturn operations: IBM SP vs. ABC.

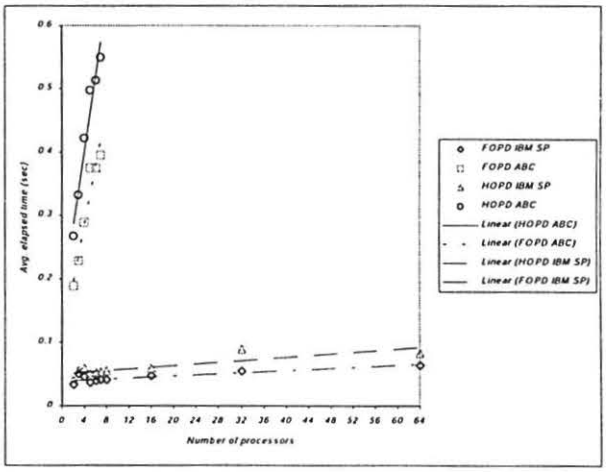


Fig. 9. Time spent in source/sink communications: IBM SP vs. ABC.

TABLE V

SUSTAINED PERFORMANCE FOR FOPD AND HOPD ON THE IBM SP

first-order post-Doppler	Flops Count	Exec. Time (sec)	Bench. Req. (Gflops/sec)	% sustained
Video to I/Q conversion	57,016,320	0.034	1.77	94.72
Array calibration and Pulse comp.	67,633,152	0.020	2.10	100.00
Doppler processing	15,728,640	0.007	0.49	100.00
Weights computation	63,700,992	0.011	1.98	100.00
Weights application	3,932,160	0.001	0.12	100.00
high-order post-Doppler				
Video to I/Q conversion	78,397,440	0.059	2.43	54.92
Array calibration and Pulse comp.	92,995,584	0.037	2.88	86.92
Doppler processing	21,626,880	0.011	0.67	100.00
Weights computation	1,074,991,104	0.178	33.33	18.09
Weights application	16,220,160	0.004	0.30	100.00