A Hitchhiker's Guide to the IBM RS/6000 SP

POE, Compiler, Libraries, Tools

Holger Holthoff
IBM RS/6000, Heidelberg
eMail: holthoff@de.ibm.com

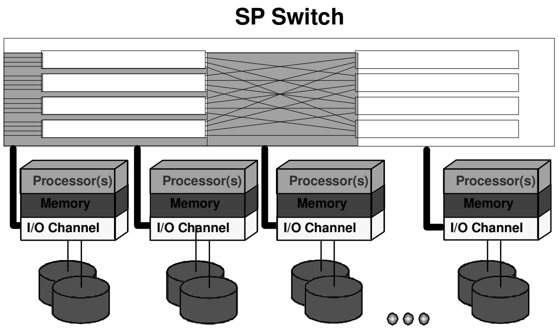
Outline

- SP hardware, AIX self-help
- Executing parallel jobs
  - Parallel Environment (POE)
  - `/usr/bin/poe`, ...
  - Tools: `xprofiler`, `vt`, debugging, ...
- Parallel enablement
  - MPI, HPF, ...
  - Compiler, Libs
  - Hunting performance bottlenecks
- Some useful references

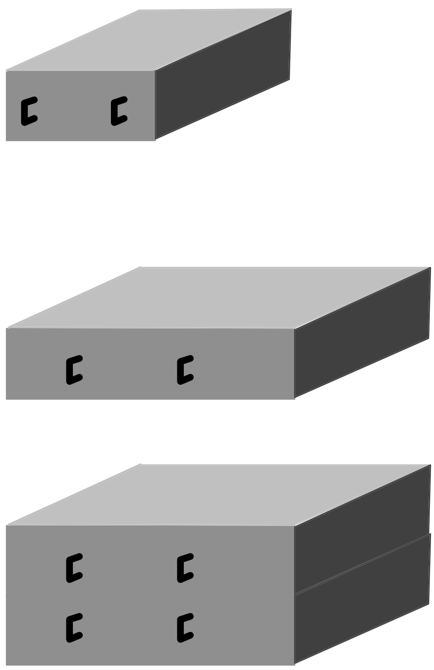
RS/6000 SP

Requirements for Scientific/Technical Computing

- Parallel jobs for "engineering reasonable" execution time
- Serial and parallel job throughput
- Expand capabilities by running industrial "grand challenge" problems



SP Nodes

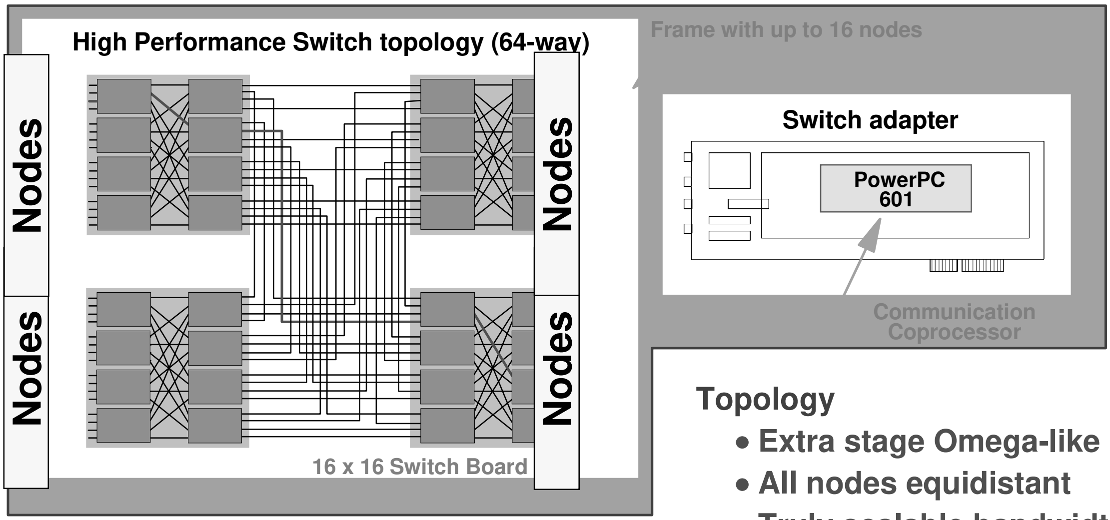


160 MHz Thin Node
160 MHz POWER2 Super Chip
64 - 1024 MB Memory
4.5 - 18.2 GB Disk
4 Micro Channel Slots

135 MHz Wide Node
135 MHz POWER2 Super Chip
64 - 2048 MB Memory
4.5 - 36.4 GB Disk
7 Micro Channel Slots

604 High Node
200 MHz 604 PowerPC - 2, 4, 6, 8-way
1800 MB/s Parallel Data Switch
64 - 4096 MB Memory
4.5 - 18 GB Disk
14 Micro Channel Slots

SP Communication Subsystem



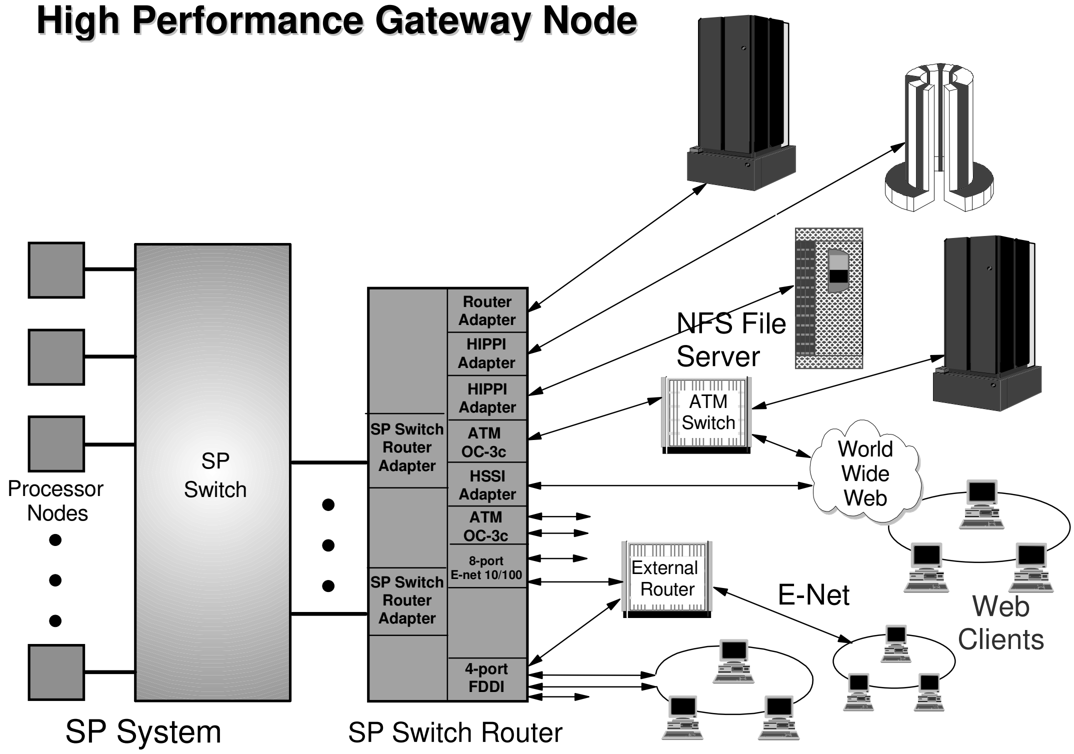
Topology
- Extra stage Omega-like
- All nodes equidistant
- Truly scalable bandwidth
- Fault resilient
- Modular

Reliability and Availability
- Redundant paths
- Fully checked
- End-to-end acknowledgment
- Automatic retry

Transparent Recovery from failures

Two Protocols that coexist
- Light-weight user mode
- IP

High Performance Gateway Node



RS/6000 CPU Architectures

- Power2
  - working horse for CAE: 64 bit floating-point unit
  - currently: P2SC 160 MHz (SPECfp95 25.8)
  - future (2H98): Power3 (64 bit, Uni and SMP, 200 -... MHz)
- PowerPC
  - allround CPU from Laptop to SP
  - currently: 166 - 332 MHz, Uni to 8-way SMP
    (F50: 4-way 332 MHz SMP, SPECfp95 12.6, SPECint95 14.4 per CPU)
- RS64A
  - 64 bit SMP data base server only
- Trend: 64 bit @ 1 GHz

CPU and Bandwidth Performance

RS/6000 Model	SP Node Type	CPU Clock Freq [MHz]	Peak [MFlop/s]	SPECfp95	Peak Memory BW [GB/sec]	Sustainable Memory BW** [MB/sec]
390	Thin	67	266	7.50	0.53	226
39H / 3CT	Thin2	67	266	9.35	1.07	451
590*	Wide*	67	266	10.4	2.13	903
591*	Wide2*	77	308	12.4	2.46	1038
	Thin3-P2SC*	120	480	16.6	1.92	1024
595*	Wide3-P2SC*	135	540	17.6	2.13	910
397*	Thin4-P2SC*	160	640	25.8	2.56	> 1038

* Four or more memory cards assumed
** Estimated values using approximate cache miss costs

AIX Help ...

- Manual pages

  `rsh <sp node> "/usr/bin/info -display ..."`

  `$LANG, $LC_MESSAGES ==> en_US`
- System administration

  `/usr/bin/smit` or `smitty` (try `F6 or "Command"`)

  see afterward: `$HOME/smit.log, smit.script`

  see also: `/usr/bin/ls*, mk*, rm*`
- AIX FAQ

  http://www.cis.ohio-state.edu/hypertext/faq/usenet/aix-faq/top.html
- "AIX Survival Guide", Andreas Siegert, Addison-Wesley

Parallel Environment (POE)

- Compiling and linking MPI programs
  - `mpxlf`, `mpcc`, ... instead of `xlf`, `cc`, ...
  - `mpi.h`, `mpif.h` in `/usr/lpp/ppe.poe/include`
- User authentication
  - see: rsh, $HOME/.rhosts, DCE/DFS, NFS, AFS, ...
- Starting a parallel job
  - `[poe] [-h] [program] [program flags] [poe options]`
  - batch queueing system (IBM LL, NQS, LSF, Codine, ...)
- User Space (us) vs. ip protocol, Resource Manager
  - monitoring POE jobs: `jm_status -Pv` or `jm_status -j` and `poestat, poekill`

Parallel Environment (cont'd)

- Environment variables and command line flags:

▸ MP_PROCS / -procs	number of program tasks
▸ MP_HOSTFILE / -hfile	name of host list file for node allocation
▸ MP_RMPOOL / -rmpool	pool for node allocation instead of hostfile
▸ MP_EUILIB / -eulib	communication protocol (us, ip)
▸ MP_CMDFILE / -cmdfile	POE command file (mpmd only)
▸ MP_PGMMODEL /	spmd (default) or mpmd
▸ MP_LABELIO / ...	labeling output of parallel tasks
▸ MP_INFOLEVEL / ...	level of message reporting
▸ MP_EUIDEVICE / ...	adapter for message passing (css0, en0, ...)
▸ ...	
▸ ...	

Parallel Environment (cont'd)

- File handling
  - NFS mounted vs. local file systems
  - POE needs at least a common name of the working directory on all nodes
  - `mcp`, `mcpgath`, `mcpscat`, `mpamddir`, `mprcp`, ...
- Distributed shell
  - `/usr/lpp/ssp/bin/dsh`
  - Kerberos authenticated shell (usually root only)
- System administration
  - File collections, SDR, ...
  - `dsmit` (distributed smit)

Parallel Environment (cont'd)

- cd $HOME

```
#!/usr/bin/ksh

export HUGO=1

export MP_RESD=yes
export MP_RMPOOL=0
export MP_PROCS=4
export MP_LABELIO=yes
export MP_STDOUTMODE=ordered
export MP_STDINMODE=none

date
poe -newjob yes << EOJ
  hostname
  env
  ulimit -a
quit
EOJ
```

Parallel Environment (cont'd)

- cd /tmp or ...

```
#!/usr/bin/ksh

export MP_RMPOOL=0
export MP_EUILIB=us
export MP_EUIDEVICE=css0
export MP_INFOLEVEL=2
export MP_NEWJOB=yes
...

/usr/bin/poe << EOJ
  mcp /u/hugo/foo.dat foo.dat
  mcp /u/hugo/a.out a.out
  a.out
  ...
quit
EOJ
```

PVMe Jobs

```
#!/bin/ksh

JOBNAME=x16
WORK_DIR=/u/holthoff/
EXE=a.out

# PVME config file ... (current node should be within hostlist)
CONFIG_FILE=$WORK_DIR/config_file
cat > $CONFIG_FILE << EOF
#
v01n17 dx=/usr/lpp/pvme/bin/pvmd3e wd=$WORK_DIR ep=$WORK_DIR
v01n18 dx=/usr/lpp/pvme/bin/pvmd3e wd=$WORK_DIR ep=$WORK_DIR
...
v01n31 dx=/usr/lpp/pvme/bin/pvmd3e wd=$WORK_DIR ep=$WORK_DIR
v01n32 dx=/usr/lpp/pvme/bin/pvmd3e wd=$WORK_DIR ep=$WORK_DIR
EOF

...
```

PVMe Jobs (cont'd)

```
...

# PVMe tuning ...
export LANG=En_US
export PVMe_DATA_INTERRUPT=no
export PVMHFN=pvmdname.$JOBNAME
export N64KBUFF=512
export PVMe_MEMCPY=power2
export PVM_EXPORT=N64KBUFF:PVMe_MEMCPY:LANG:PVMe_DATA_INTERRUPT

PVMD_PATH=/usr/lpp/pvme/bin/pvmd3e
PVMC_PATH=/usr/lpp/pvme/bin/pvme

# Start PVMe daemon ...
$PVMD_PATH $CONFIG_FILE &
```

PVMe Jobs (cont'd)

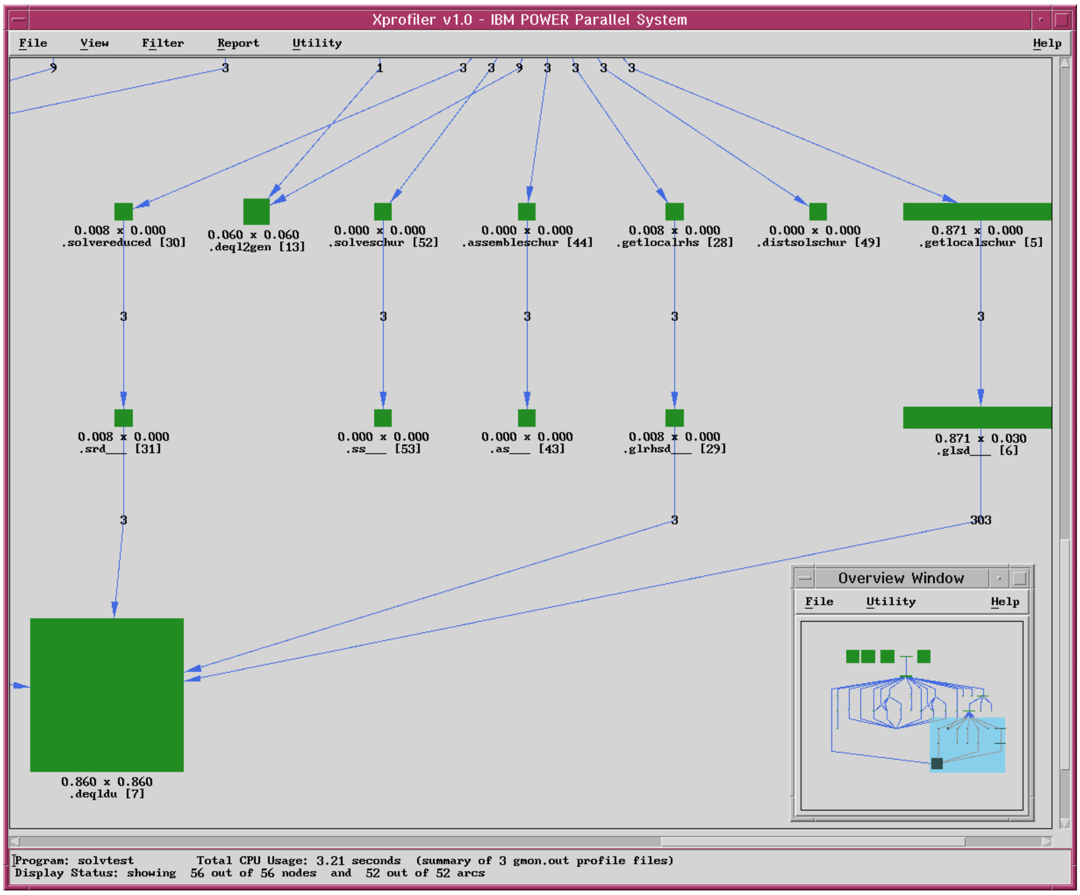
```
...
sleep 10
ps -ef | grep "$PVMD_PATH" > /dev/null
if [ $? -eq 0 ] ; then
    echo PVM daemon launched
else
    echo Error: PVM daemon failed
    exit 16
fi

# Start job ...
/usr/bin/time $WORK_DIR/$EXE < $WORK_DIR/infile \
                             > $WORK_DIR/outfile 2>&1

# Halt PVMe ...
$PVMC_PATH << EOF
halt
EOF
```

Tools - Profiling

- prerequisite:
  - compiling with -pg -g
  - linking with -pg
  - program execution yields gmon.out (sequential jobs) or gmon.out.<n> (parallel POE jobs, <n> = process ID)
- standard: `prof`, `gprof`, `tprof`
- Xprofiler:
  - `xprofiler a.out gmon.out ... [options]`
  - X11 based frontend for profiling files
  - subroutine calling tree with timing information, ...



Tools - Profiling (cont'd)

- **vt** (Visualization Tool):
  - X11 based frontend for PE tracefiles
  - invoke parallel program with `export MP_TRACELEVEL=9` or `-tlevel 9`
  - `vt -tfile foo.trc`
  - categories: communication, computation, system, network, disk, ...
  - warning: trace files become huge very quickly (see also: `MP_TMPDIR, MP_TRACEDIR, MP_TBUFFSIZE, MP_TTEMPSIZE, MP_TBUFFWRAP`)
- MPI/MPL pmarray extension (Program Marker Array)
- LMPI -- post mortem analysis (MPICH upshot)
  - LRZ Munich (http://www.lrz-muenchen.de/services/software/parallel/lmpi/)

Tools - Profiling (cont'd)

- cd $HOME/...

```
#!/bin/ksh

export MP_PROCS=4
export MP_RESD=yes
export MP_RMPOOL=0
export MP_EUILIB=us
export MP_HOSTFILE=NULL
export MP_INFOLEVEL=2
export MP_TRACELEVEL=9
export MP_PMLIGHTS=3

rm -f mmp.trc
pmarray &
sleep 3

/usr/bin/time poe mmp
# pedb mmp
vt -tfile mmp.trc
```

Tools - Debugging

- prerequisite: compiling with `-g` (source code information)
  `-qfullpath` preserves absolute path name of source files
- **dbx**: sequential programs, ASCII interface
- **pdbx**: parallel programs, ASCII interface, based on dbx
- **xldb**: sequential programs, X11 interface
- **pedb**: parallel programs, X11 interface
  syntax:
  <debugger> [[program] program options][poe options][debugger options]
  -a PID: attach to a running (POE) process
  -h: write debugger usage to STDERR
- **TotalView:** third party debugger available on several platforms

Parallel Enablement: a simple MPI example

```
#include "mpi.h"
#include <math.h>
int main(int argc, char *argv[])
{
    int n, myid, numprocs, i, rc;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x, a;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    while(1) {
        if (myid == 0) {
            printf("Enter the number of intervals: (0 quits)\n");
            scanf("%d", &n);
        }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

Parallel Enablement: a simple MPI example (cont'd)

```
        if (n == 0)
            break;
        else {
            h = 1.0 / (double) n;
            sum = 0.0;
            for (i=myid+1; i<=n; i+=numprocs) {
                x = h*((double)i - 0.5);
                sum += (4.0 / (1.0 + x*x));
            }
            mypi = h*sum;
            MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE,
                       MPI_SUM, 0, MPI_COMM_WORLD);
            if (myid == 0)
                printf(
                  "pi is approximately %.16f, Error is %.16f\n",
                  pi, fabs(pi-PI25DT));
        }
    }
    MPI_Finalize();
}
```

Parallel Enablement: MPI

- Strategy: domain decomposition vs. parallel solvers (PESSL, WSSMP, Scalapack, ...)
- Tip: "incremental" distributed memory parallelization
  - in case of moderate or low memory requirements
  - create a simple hostless MPI main program to run the same executable on all nodes
  - implicit duplication of all variables on all nodes
  - step by step parallelization and reduction of memory consumption
- My favourite introduction:
  - "Using MPI", Gropp, Lusk, Skjellum, MIT Press

Parallel Enablement: APIs besides IBM MPI-1.1

- Message Passing ("MPMD", "MPP")
  - Public domain MPI (MPICH, LAM, ...), PVM(e), MPL, PARMACS, PVMPI, ...
  - MPI-IO, MPI-2: PIOFS, GPFS, ...
  - One-sided communications (put, get)
- Data Parallel ("SIMD")
  - HPF (xlhpf, APR/FORGE), ANSI Data Parallel C (DPC, Pacific-Sierra Research Inc.)
- Shared Memory ("SMP")
  - XLF-SMP, OpenMP, POSIX Threads, ...
- Virtual Shared Memory
  - TreadMarks, Myrias/PAMS, Piranha, Linda, IBM/ABC++ ...
- Hybrid MPP/SMP Architectures
  - pure MPI
  - mixture of MPI and SMP autoparallelization (or PThreads)

Compiler: xlf (F77/90)

- F77/F90 with proprietary extensions
- Compiling and linking
  - `xlf`: F90, 'fixed format'
  - `xlf90`: F90, 'free format'
  - `xlf_r`, `xlf90_r`: F90, thread-safe, SMP programming
  - `mpxlf,mpxlf90,`
    `mpxlf_r,mpxlf90_r`: F90, MPI or MPI/SMP
- xlf 5.1: F90 Pthreads library module (f_pthread)
- Configuration file: `/etc/xlf.cfg`

Compiler: xlf (F77/90) (cont'd)

- Options for performance optimization

`-O, -O3, (-qstrict)`	enables the level of optimization
`-qarch=<arch>`	architecture (pwr2, 604, ppc, com, ...)
`-qtune=<arch>`	hardware implementation (pwr2, 604, ...)
`-qhot`	high order transformations
`-qipa`	interprocedural analysis
`-qfloat`	details `real*4` floating-point processing
`-Q`	inlining

- Misc. flags

`-pg`	profiling (compile and link step)
`-g`	include source code information
`-C`	check array bounds
`-bmaxdata`	data segments (>256 MB)

Compiler: xlhpf (HPF)

- High Performance Fortran Compiler
- HPF subset
- Compiling and linking:
  - `xlhpf`: F90, 'fixed format'
  - `xlhpf90`: F90, 'free format'
- Configuration file: `/etc/xlhpf.cfg`
- HPF and MPI within one program allowed

Compiler: xlf Version 5.1 (SMP)

- Since version 5.1: SMP autoparallelization
- Not yet fully OpenMP compliant
- Flags:
  - `-qsmp`: autoparallelization (w/suboptions)
  - `-qnosave`: always for SMP codes
  - `-qreport`: report of compiler actions
- Environment variables control program execution
  - `XLSMPOPTS`: e.g. `parthds=<n>` (number of parallel threads)
  - `XLFRTEOPTS`

Compiler: xlC (C/C++)

- C, C++ and proprietary extensions
- Compiling and linking
  - `xlc`: C
  - `xlC`: C, C++ (extensions .c oder .C)
  - `xlc_r, xlC_r`: C, C++, thread-safe, SMP programming
  - `mpcc,mpCC, mpcc_r,mpCC_r`: C, C++, MPI or MPI/SMP
- Configuration file: `/etc/xlC.cfg`

Compiler: xlC (C/C++) (cont'd)

- Options for performance optimization:

-O, -O3, (-qstrict)	enables the level of optimization
-qarch=<arch>	architecture (pwr2, 604, ppc, com, ...)
-qtune=<arch>	hardware implementation
-qipa	interprocedural analysis
-qfloat	details `float` floating-point processing
-Q	inlining

- Misc. flags:

-pg	profiling (compile and link step)
-g	include source code information
-bmaxdata	data segments (>256 MB)

Tools - Preprocessors, Editor

- **KAP**:
  - proprietary preprocessor to optimize Fortran or C code
  - e.g.: `xlf -Pk <other compiler flags> ...`
- **VAST**:
  - similar to KAP
  - e.g.: `xlf -Pv <other compiler flags> ...`
- **SDE Workbench, LPEX editor**
  - programmable live parsing editor
  - syntax highlighting, make, error handling
  - `lpex <filename>` or CDE drag&drop

Libraries

- Engineering and Scientific Subroutine Library (ESSL):
  - Vector and matrix operations, BLAS
  - Linear algebraic equations, FFT, etc.
- Parallel ESSL (PESSL):
  - Subset of ESSL functions
  - Parallelized for MPI, HPF, and SMP
- Mathematical Acceleration SubSystem (MASS):
  - Acceleration of mathematical intrinsic functions (sin, sqrt, exp, ...)
  - Requires just relinking (`-lmass -lmassvp2 -lm`)
  - Vector library (vsin, vcos, ...) needs little programming work
    vexp(y,x,n): y(i)=exp[x(i)], i=1,n

Hunting Performance Bottlenecks

- Monitor global system resources
  - Answer basic questions
    CPU bound, Memory bound, I/O bound, or Network bound
  - Least details to interpret
- System wide resource monitoring tools
  - ps: e.g. `ps au` or `ps vg` or `ps -el`
  - vmstat: e.g. `vmstat 3 5`
  - svmon: e.g. `smvon -G`
  - iostat: e.g. `iostat -d 5 2`
  - sar: e.g. `sar -P ALL 5 3`
  - AIX Performance Toolbox: 3D monitor `xmperf`

Hunting Performance Bottlenecks (cont'd)

- Program performance
  - Time measurements
    Fortran: `rtc()`
    C: `gettimeofday()`
    job level: `time, timex, /usr/bin/time`
  - Xprofiler
  - Program visualizer (PV)
  - XCacheRS6K
  - ...

Public Domain Hardware Performance Monitor

- rs2stat, rs2mon

  performance of whole node

  `rs2stat [-w] [delay [iter]]`

  `rs2mon [delay [iter]]`
- rs2hpm

  monitoring a single program, but depending on system load

  `rs2hpm [-S|-E -f` *`prog`*`.hpm [-z]] [-s|-u]` *`prog`*
- sp2flops

  `sp2flops [hostlist]`
- monitor

  `monitor [-top | -all]`

Single Node RISC Performance Tuning

- General Strategy
  - Generate data locality, minimize data transfer, improve computational density → BLAS-3 type operations
  - Use optimized libraries (ESSL, PESSL, MASS, ...)
- Practical Hints
  - Avoid unnecessary cache misses, introduce **cache reuse**
    - ➡ access of vector elements with stride 1 if possible
    - ➡ rearrangement of formal array dimensions (avoid powers of 2)
    - ➡ rearrangement of loops in order to perform as many as possible operations on the data before restoring them (blocking of matrix operations, ...)
  - Unrolling of loops to overlap computation and data movement
  - Avoid unnecessary divisions
  - Encapsulation of critical code segments into subroutines
  - ...

Single Node RISC Performance Tuning (cont'd)

- C
  - Reducing pessimistic aliasing assumptions in C programs
    `#pragma disjoint(a,b)`
    `-qignprag=disjoint`
  - Functions with no side effects
    `#pragma isolated_call(func1)`
  - The ansialias option
    `-qansialias`
  - Use array notation instead of pointers to handle strings
  - Prefer built-in string functions over intrinsics: `<string.h>`
  - Never `malloc()` more memory than needed
  - Do not use `calloc()` unless required
  - ...

Fine Tuning AIX (experts only)

- vmtune
  - modifies VMM behaviour
    page stealer (minperm/maxperm)
    free list (minfree/maxfree)
  - modifies file system, LVM, and paging parameters
  - must be re-run at every reboot
- schedtune
  - modifies/disables memory load control
  - modifies CPU scheduling algorithms
  - modifies CPU time slices
  - restores defaults
  - must be repeated at every reboot

IBM Documentation

- IBM Parallel Environment for AIX:
  - Operation and Use, Volume 1 & 2
  - MPI Programming and Subroutine Reference
  - Messages
  - Hitchhiker's Guide
- XL Fortan, C Set++, XL HPF for AIX:
  - User's Guide and Language Reference
- Optimization and Tuning Guide for Fortran, C, and C++

Online: http://www.rs6000.ibm.com/resource/aix_resource/sp_books/
file:/usr/lpp/<compiler>/postscript

Surfing the web

RS/6000 Home Page:	http://www.rs6000.ibm.com
SP Hard- and Software:	http://www.rs6000.ibm.com/sp.htm
SP Online Docs:	http://www.rs6000.ibm.com/resource/aix_resource/sp_books/
MASS Lib:	http://www.rs6000.ibm.com/resource/technology/MASS/
Hardware Performance Monitor:	http://www.csc.fi/~jmaki/
PD Software as SMIT install images:	http://www-frec.bull.com/
AIX FAQ:	http://euch6h.chem.emory.edu/services/aix-faq/ http://www.cis.ohio-state.edu/hypertext/faq/usenet/aix-faq/top.html
Argonne SP User Group:	http://www.mcs.anl.gov/Projects/sp/users-group.html
Cornell:	http://www.tc.cornell.edu/UserDoc/SP/
ASCI Blue:	http://www.llnl.gov/asci/
Uni Karlsruhe:	http://www.uni-karlsruhe.de/~SP2/ http://www.uni-karlsruhe.de/~AIX/

Surfing the web (cont'd)

Fortran:	http://www.software.ibm.com/ad/fortran/ http://www.software.ibm.com/ad/fortran/xlfortran/cray.htm
MPI:	http://www.mcs.anl.gov/mpi/