# Basic Profiling Tools and Hardware Counter Usage

Get started with some standard tools on blizzard

Hendryk Bockelmann, DKRZ



### Access to performance analysis tools

- standard profiling available (like -p and -pg compiler flags)
- source /usr/lpp/ppe.hpct/env\_sh for IBM tools (libmpitrace and HPM) to set IHPCT\_BASE and paths
- use modules for vampirtrace, scalasca and vampir(server)

#### module environment:

- module av: shows all modules available
- module add/rm: adds, removes modules
- on blizzard use: scalasca/1.4.2rc3-aixpoe-ibm, vampirtrace/5.13rc1-aixpoe-ibm (latest versions), maybe go back to stable versions ...
- on lizard use: vampir/vampirclient-7.5.0



## Sequential profiling with prof

- displays object file profile data
- for each text symbol in object file the percentage of execution time, number of times that function was called, and the average number of milliseconds per call is shown
- easy to use: compile and link with additional -p option, run your program as usual
- produce profiling information:

prof ./foo.x -m mon.out > foo.mon.rpt

output is flat profile of called functions, even MPI calls are profiled at lowest level (ie. \_lapi\_\*)



## Seq profiling with prof (example)

4/33

File Edit View Search	Terminal	Help				
Name	%Time	Seconds	Cumsecs	#Calls	msec/call	
. lapi shm dispatche						
. lapi dispatcher(un	9.8					
.LAPI_Msgpoll	8.6		74.91			
mo_diffusion_NMOD						
mo_adpo_NMOD_ocad		10.97				
. mo_adpo_NMOD_ocad	5.0	10.1				
mo_baroclinic_NMO			115.12			
mo_iolist_NMOD_av		6.58	121.70			
mo_adpo_NMOD_ocad		6.44	128.14			
is_yield_queue_emp	3.0		134.34			
mo_ocjitr_NMOD_oc		5.56	139.90			
mo_boundsexch_NMO		4.98	144.88	18939	0.2629	
mo_ocmodmom_NMOD_		4.53	149.41			
<pre>mo_adisit_NMOD_po</pre>		3.99				
.do_ccl			156.95			
mo_diagnosis_NMOD			159.86			
<pre>lapi_memcpy(void(*</pre>		2.07	161.93			
mo_adpo_NMOD_adve		2.00	163.93			
mo_ocvisc_NMOD_oc			165.92			
mcount	0.9		167.72			
mo_iolist_NMOD_fi	0.9		169.50			
<pre>mo_ocean_vertical</pre>	0.8		171.12			
. mo_ocjitr_NMOD_oc	0.8	1.62	172.74			
.ocuad			174.13			
mo_commo1_NMOD_lw			175.489	7275384	0.0000	
mo_adpo_NMOD_ocad		1.34	176.82			v



Hendryk Bockelmann, DKRZ

## Sequential profiling with gprof

5/33

- in addition to prof, gprof shows the call-graph data and not only flat profiles
- easy to use, available on most systems: compile and link with additional -pg option, run your program as usual
- produce profiling information:

gprof ./foo.x gmon.out > foo.gmon.rpt

 drawback: profiling support is added by the compiler, so if you wish to obtain profiling information from any shared libraries, you need to also compile them with -pg (therefore MPI calls \_lapi\_\* appear spontanously)



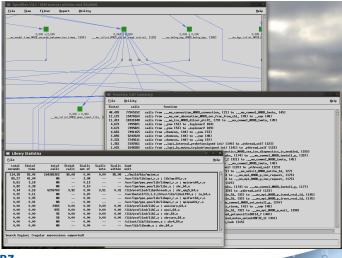
## Seq profiling with gprof (example)

1.5 8.80 8.46				om [1]	estep comput		
6.60		1/1			ime invariar		
				tro	NMOD_corre	ct_zo_ [315]	
							_t1st_ [742]
						rt_[882]	
		🖬 boc	kelman	n@tide: ~			
		File Ed	lit View	Search	Terminal He	lp.	
		1					
		6165					
		(4)					
		/1					
							.per timestep computation [3]
		151				0 voi40 model di	. mo adpo NMOD advection [5]
		(3)				816/816	mo_adpo_NMOD_ocadpo_trf_[6]
				0.68		48/48	mo adpo NMOD ocadpo base [63]
							mo boundsexch NMOD bounds exch_halo_3d [24]
		_					. mo adpo NMOD advection [5]
		[6]					
1 A HELL							. mo_adpo_NMOD_ocadpo_trf_v_[12]
	A DECEMBER OF STREET,						. mo adpo NMOD ocadpo trf u [13]
		CONTRACT OF THE OWNER OF THE OWNE					<pre>. mo adpo NMOD ocadpo trf z [19] . mo boundsexch NMOD bounds exch halo 3d [24]</pre>
100							
THE PART		6.65					
		[7]					
				0.00	0.00		.pthread mutex lock [313]
					0.00		.pthread mutex unlock [399]
	0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,0	0.00         1.11           0.00         0.10           0.00         0.410           0.00         0.410           0.00         0.410           0.00         0.411           0.00         0.411           0.00         0.401           0.00         0.400           0.00         0.400           0.00         0.400           0.00         0.400           0.00         0.400           0.00         0.400           0.00         0.400           0.400         0.400           0.400         0.400           0.400         0.400           0.400         0.400           0.400         0.400           0.400         0.400	0.00         1.11         1.1           0.00         0.01         1.1           0.00         0.01         1.1           0.00         0.01         1.1           0.00         0.01         1.1           0.00         0.01         1.1           0.00         0.01         1.1           0.00         0.01         1.1           0.00         0.01         0.01           0.00         0.00         0.01           0.00         0.00         0.01           0.00         0.00         0.01           0.00         0.00         0.01           0.00         0.00         0.01           0.00         0.00         0.01           0.00         0.00         0.01           0.00         0.00         0.01           0.00         0.00         0.01           1.5         0.00         100.57	0.00 1.11 1/1 0.00 0.01 1/1 0.00 0.01 1/1 0.00 0.01 1/1 0.00 0.01 1/1 0.00 0.01 1/1 0.00 0.01 1/1 0.00 0.00 0.00 0.00 1/1 0.00 0.00 0.00 0.00 1/1 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.0	0.00         1.11         1/1         .vrite", 0           0.00         0.18         1/1         .rest to 0           0.00         0.018         2/1	0.00         1.11         1/1         -write restrict file           0.00         1.01         .resd user file         1/2           0.00         0.01         1/1         .set use file         1/2           0.00         0.01         2/2         -set use file         1/2           0.00         0.01         2/2         -set use file         1/2           0.00         0.01         2/2         -set use file         1/2           0.00         0.00         .set use file         1/2         .set use file           0.00         0.00         0.00         0.00         0.00         0.00           0.00         0.00         0.00         0.00         0.00         0.00           0.00         0.00         0.00         0.00         0.00         0.00           0.00         0.00         0.00         0.00         0.00         0.00           0.13         0.00         0.00         0.00	0.00         1.11         1/1

DEUTSCHES

Hendryk Bockelmann, DKRZ

### Use xprofiler for gmon.out



DKRZ DEUTSCHES KLIMARECHENZENTRUM

Hendryk Bockelmann, DKRZ

### Processor usage report with tprof

- tprof charges processor time to object files, processes, threads, subroutines (user mode, kernel mode and shared library) - using AIX trace utility started in background
- advantage: subroutine-level profiling without modifying executable programs (no recompile, relink needed)
- modify batch script:

trcstop 2>/dev/null
tprof -usz -p <binary> -x poe /path/to/binary

- profiles all processes on system (use node not\_shared)
- output of tprof is very flat, but shows all needed information
- only one node will be profiled (the one on which poe is started!)



## System profiling with tprof (example)

File Edit View Search Terminal Help

Total Samples = 961911 Traced Time = 238.54s (out of a total execution time of 238.54s)

Process				FREQ		Kernel	User	Shared			
-tests/mpio	m-aprof/bu	ild/bin/r	npiom.x	540 7	750755	6548	497002	247204	1		
PID-1				207 2	210913	210671	127	115	Θ		
/etc/pmdv5								11			
/usr/bin/po											
/usr/bin/sh											
/etc/pm set	affinity										
/usr/bin/tr	cstop										
======											
Total					961911	217427	497135	247344			
Process		PID	ckelm <del>TID</del>	Total		el Use	er Shar	ed Othe			
		====		Search	- R.H.H.H.		;= =====	== ====	-		
of/build/bi			20055039	2350		68 175			0		
of/build/bi			40174389	23504		65 1624			0		
of/build/bi			80085989	23503		69 1762					
of/build/bi			68879211	23499		64 1660			0		
of/build/bi			80478743	23494		53 1788			9002		
of/build/bi			12517889	23493		62 1774			<b>0</b> 127		
of/build/bi	n/mpiom.x	5833004	56164747	23492	2 1	57 1636	58 69	57	<b>0</b> 5		



### System profiling with tprof (example)

10/33

File Edit View Search Terminal Help

Total Ticks For Al	l Processes (USEF	() <sup>2</sup> ≠° <b>4971</b> 3 7094	35 °			
User Process <sup>89</sup> 2350					Address	
profiling-tests/mpio	n-gprof/build/bin	/mpiom.x	497002	51.67	100000240	84ba74
PID 1 12517889 2349				0.01	100079e0	f0c38d0d
/etc/pmdv54747 2349				0.00	1000001f8	ef4a8
/usr/bin/poe			1	0.00	1000001f8	e3a88

Profile: /work/k20200/k202082/profiling-tests/mpiom-gprof/build/bin/mpiom.x

Total Ticks For All Processes (/work/k20200/k202082/profiling-tests/mpiom-gprof/build/bin/mpiom.x) = 497002

Subroutine	Ticks		Source	Address	Bytes	
ffusion NMOD octdiff trf	86799	9.62	src/mo diffusion.f90	623c00	d3a0	
o adpo NMOD ocadpo trf v	45991		n//src/mo_adpo.f90	67f760	3780	
o adpo NMOD ocadpo trf u	43318		n//src/mo_adpo.f90	682 xe0	3f20	
o adpo NMOD ocadpo trf z	36246		n//src/mo_adpo.f90	686e00	4940	
o baroclinic NMOD occlit			rc/mo baroclinic.f90	64ad60		
glist_element_accumulate			/src/mo_iolist.f90	56fde0		
o ocjitr NMOD ocjitr trf			/src/mo_ocjitr.f90	69fae0	5940	
.carchm	18691		<pre>rc_hamocc/carchm.f90</pre>	713040	4900	
NMOD_bounds_exch_halo_3d_	15508		<pre>rc/mo_boundsexch.f90</pre>	116780	9620	
o_ocmodmom_NMOD_ocmodmom_	14818		/src/mo_ocmodmom.f90	65be60	d400	
.ocprod_	13556		rc_hamocc/ocprod.f90	6ff240	fd80	
<pre>mo_adisit_NMOD_potrho_1d_</pre>	13028		/src/mo_adisit.f90	5de5a0		
<pre>fusion_NMOD_octdiff_base_</pre>	12944	1.35	<pre>src/mo_diffusion.f90</pre>		74e0	



## System profiling with tprof (example)

File Edit View Search Terminal Help						
Total Ticks For All Processes (SH-LIBs) = 2	47344					
8 8 7 0 Shared Object 1 1 0	1 <sup>0</sup> Ticks		Address Bytes			
			====== =====			
/usr/lib/liblapi <sup>8</sup> r.a[liblapi644r.o] <sup>97135</sup> 2473	218930		90000002380480	15f734		
usr/lpp/ppe.poe/lib/libmpi r.a[mpicore64 r.o]	25859	2.69	9000000020a2c00	27d15c		
/usr/lpp/ppe.poe/lib/libc.a[shr_64.o]	1666	0.17	900000000000900	3108fc		
/usr/lib/libpthreads.a[shr_xpg5_64.0]			9000000005e0200			
usr/lpp/ppe.poe/lib/libmpi_r.a[mpifort64_r.o]			900000001b17fc0			
/usr/lpp/ppe.poe/lib/libmpi_r.a[mpipoe64_r.o]			90000000206f280	271b7		
/usr/lib/libtrace.a[shr.o]			d1d5e280 2e17c			
ibm/xlf/13.1.0.10/usr/lib/libxlf90.a[io_64.o]			90000000504cf00			
/usr/lib/libc.a[shr_64.o]			900000000000900	3108fc		
/usr/lib/libpthreads.a[shr_xpg5.o]			d04ea180 3157b			
/usr/lib/libc.a[shr.o]			d0118680 31404c			
/usr/lib/libptools.a[shr.o]			042b2280 33228			
/usr/lib/libs.a[shr_64.0]			900000000000900			
/usr/lib/IbBaseLib.a[shr_64.0]			9000000007f6c00			
/usr/lib/libodm.a[shr_64.o]			900000003ebe80			
/usr/lib/libllrapi.a[shr.o]			d2e09260 6d9dfc			
/usr/lib/libhal_ib.a[hal_ib64.o]			9000000282d600			
/usr/lpp/ppe.poe/lib/libc.a[shr.o]			d0118680 31404c			
/usr/lib/libllrapi.a[shr_64.o]			900000000a51ce0			
/usr/lib/libpnsd.a[shr_64.o]			9000000274cd80	c8030		
/opt/freeware/lib/ganglia/modcpu.so			d05c6128 4b0f			
/usr/lib/libC.a[ansicore_64.o]		0.00	9000000005a3400	f850		



## Summary of \*prof

src change, output given:

- gprof: standard, supported profiling tool on many UNIX systems - no need to learn new stuff
- tprof: collects data with no impact on execution time, works on optimized binaries without any need for recompilation - prof, gprof might have overhead, might not work on optimized binary, need recompile, relink
- prof, gprof: provide subprogram profiling, exact counts of number of times every subprogram is called - tprof does not
- gprof: provides call graph prof, tprof do not



## Summary of \*prof

what data is given:

- all \*prof obtain processor consumption estimates for each subprogram by sampling program counter of user program
- tprof collects processor usage information for whole system prof, gprof get profiling information for single program & time in user mode

treatment of MPI data:

- all \*prof tools give flat information on CPU-time used
- all MPI-stuff is hidden, each process gives its own profile (no correlations can be drawn)
- $\rightarrow$  MPI profiling needed



## MPI profiling

14/33

What we know by now:

- time spent in user subroutines
- time spent in MPI-lib

What we do not know by now:

- which MPI routines take the time
- correlations between MPI-tasks (send/receive/wait)
- $\rightarrow$  tools are needed (this is why you are here)
  - libmpitrace
  - scalasca
  - vampir(trace)



## HPCTlib: libmpitrace

- calls to MPI routines are intercepted by library functions
- MPI profile data of MPI routines can be collected during a programs execution
- on MPI\_Finalize, data is gathered and profile data is written
- compile your application with -g to enable mapping of performance information back to the application source
- b link your application with -L/usr/lpp/ppe.hpct/lib64 -lmpitrace
- profile data written to mpi\_profile.<rank> and XML file mpi\_profile\_<rank>.viz for visualization



### libmpitrace: profile

- default: data only for rank 0 and min, max, avg rank set envVar OUTPUT\_ALL\_RANKS=yes for more
- mpi\_profile.<rank> are human readable

PI Routine: coll				
IPT Commesizees a			and that reares	 Dinformation about the call
PI Commerank to 1				
MPI Isendcation w				
MPIaIrecv written				4le <rank>.viz for visuali:</rank>
	36600			
MPI Waitle data or	nlv for min. <b>49031</b> 0		0can be mo <b>30;13</b>	8 by environment variable (
MPI Waitall <rank></rank>	viz files wit <b>204</b>		0t III) 0.29	
MPI Bcast			0.07	
MPI Barnierso ava:			0 view sing0:00	
		647699.		
MPI aRéduceor a sei			2possible 10:86	
MPIaReduceor a se MPI_Allreduceot be	lected portion c2 e controlled 1155	Sthe program54: Se API, it 431:	2possible 10:86	
total communication total elapsed time	lected portion 02 a controlled 1155 on time = 47.101 g age of :=0116.732 ributions:	Sthe program54: Se APT, it 431: seconds. seconds.e ===	2possible 10.86 5llected th2.76	
API Reduce to b API_Allreduce to b total communication total elapsed time dessage size distr and time is not en API_Send	<pre>lected portion of2 e controlled 11555 on time = 47.101; gp of = 116.732 "ibutions: nugh: getting count #calls 6386 31</pre>	<pre>5 he program54: 5 APT, 1t 431; seconds. seconds.e ==== avg. bytes 83653.2 1003837.9</pre>	2possible 10:86 5llected t 2:76 time(sec) 0.362 0.015	4h MPI Trace API 17out the entire execution (
API Reduce to b API_Allreduce to b total communication total elapsed time dessage size distr and time is not en API_Send	<pre>lected portion 02 e controlled 11555 on time = 47.101 : e po of = 116.732 "ibutions:</pre>	<pre>she program54: shaPl, it 431; seconds. seconds. seconds. avg. bytes 83653.2 1003837.9 avg. bytes</pre>	200551blc 10.86 51lected t 2.76 time(sec) 0.362 0.015 time(sec)	4h MPI Trace API 17out the entire execution (
API Reduce of a se API Allreduce of b total communication total elapsed time Aessage size distr dessage size distr	<pre>lected portion of2 a controlled 11555 on time = 47.101; gp of = 116.732 "ibutions: nugh: getting count #calls 6386 31 #calls 158192</pre>	<pre>5 he program54: 5 APT, 1t 431; seconds. seconds.e ==== avg. bytes 83653.2 1003837.9</pre>	2possible 10:86 5llected t 2:76 time(sec) 0.362 0.015	4h MPI Trace API 17out the entire execution (



### libmpitrace: profile

17/33

#### view mpi\_profile\_<rank>.viz files with peekperf

DATA VISUALIZATION WINDOW			
mpidata			im_gather.ts0 mo_jolat.ts0 mo_forcing.f90 mo_model_time.ts0 mo_boundsexch.f90 mo_grid.f90
abel	Count WalCloc	Transfered By	386 I> Define and read namelist FORCCTL
÷0			387 SUBROUTINE read_namelist_forcctl(model_start_time, io_in_forcctl, ierror)
SUMMARY			388
-MPI_Alireduce	11555 2.10527	4.98579++06	
-MPLBarrier	2 0.008804		390 INTEGER, INTENT(in) :: io_in_forecti
-MPI_Bcast	231 2.11603	2.340720+07	391 INTEGER, INTENT(OUT) : lettor
-MPI_Comm_rank	2 20-06	0	392
-MPI_Comm_size	1 1e-06	0	393 NAMELIST /forectl/ cforedata, foreing_frequency, write_foreing, 8.
-MPL Irecv	196124 0.245131	1.37570+09	394 #Indef NOCDI
-MP[]pand -MP[ Recy	198169 1.10373 207 2.79786		395 Itime_interp_forcing, lspat_interp_forcing, 8.
-MPL_Recv -MPL Reduce	207 2.79786 25 0.000906	1.83225e+07	396 luse_model_time, forcing_periodicity, 8.
-MPL_Reduce -MPI Sendrecy	29290 0.173451		397 #endf
-MPL Salutov	392248 6.83888	0	398 [periodic_forcing, &
-MPI Watal	2045 10,2707	0	399 idebug_forcing, iditf_runoff_grid, forcing_start_time
- mo boundsexch/90	2040 102100	0	400
mo diagnosis /90			401 I Set default values
mo forcing /90			402 cforodata = 'OMP' I forcing data set
- mo forcing NMOD read namelist forcot (mo forcing/90)			403 forcing_trequency = 88400_dp I forcing data period (sec)
HMPI Boast 420	1 2.5e-05	20	404 Write_forcing = FALSE. I write interpolated forcing fields
-MPI Boast 421	1 6e-06	8	405 #Indef NOCDI
-MPI Boast 422	1 7e-06	4	406 Itime_interp_forcing = .FALSE. I perform time interpolation
-MPI Boast 423	1 7e-06	4	407 Ispat_Interp_forcing = .FALSE. 1 perform spatial interpolation
-MPI Boast 424	1 6e-06	4	408 kse_model_time = .TRUE. I use model time axis to assign leap years
-MPI Boast 425	1 5e-06	4	409 forcing_periodicity = -999 1 number of available forcing years
-MPI_Bcast_426	1 6e-06	4	410 #endf
MPI_Bcast_427	1 0.000105		411 [periodic_forcing = .FALSE.   reposition forcing files to initial point at turn of year
-MPI_Bcast_428	1 4.4e-05	4	412 Idebug_forcing = FALSE. I turn on debug mode
-MPI_Bcast_429	1 6e-06	4	413 kitt_runoff_grid = .TRUE. I differing original grid for runoff data
mo_grid.f90			414 forcing_start_time = model_start_time   forcing start year
mo_iolist.f90			415
mo_model_time.f90			416   Read namelist forocti
mo_mpi/90			417 IF [ p_pe == p_io ] READ(io in forcet/forcet/costate/error)
mo_parallel.f90			418
mo_parallel_dlags./90			419 I Broadcast external namelist settings
mo_ppm_gather190			420 CALL p_bcast(cforodata, p_io)
- mo_tracer.txu - moiom.t50			421 CALL p bcast(forcing frequency, p io)
+ read namelst.f90			422 CALL p_bcast(twite_forcing, p_jo)
T. LOOP THOM MODEL ON A			423 CALL p_bcast()time_interp_forcing, p_io)
			424 CALL p_bcast(lspat_interp_forcing, p_io)
			425 CALL p_bcast(luse_model_tme, p_io)
			425 CALL p_bcast[fording_periodicity, p_jo]
			427 CALL p beast/jent/dc fireing p by 428 CALL n heastfildehun fording n kit
		(a.14)	428 CALL n heastlidebus forcing n. kl



### libmpitrace: customization

8/33

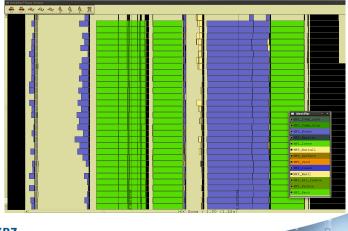
- OUTPUT\_ALL\_RANKS=[yes|no]: show all results
- TRACE\_ALL\_EVENTS=[yes|no]: do tracing or profiling
- MAX\_TRACE\_RANK=#: maximum of traced ranks
- TRACEBACK\_LEVEL=#: useful for nested MPI-calls within other functions/libraries
- manual tracing for selected portion of the program through API
- profiling data cannot be controlled by the API always collected throughout the entire execution of the program

 $\rightarrow$  single\_trace\_0 outputfile contains tracing information from MPI ranks for which tracing was enabled



### libmpitrace: trace

#### view single\_trace\_0 file with peekview



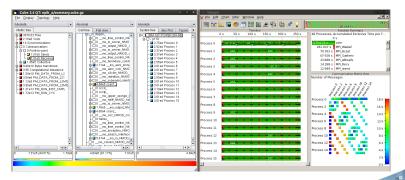


Hendryk Bockelmann, DKRZ

## Summary on basic tools

20/33

We have profiles of user subroutines and MPI calls by at least 2 different tools - one would be better Additionally, the visualization could be better ...





### If CPU time is not enough ...

Knowing hotspots of the application is only the beginning:

- why does some routine take so much time?
- why does MPI take so much time?

Answers: not trivial but often related to one of these issues

- ▶ parallelization aspects (imbalance, race-conditions, etc.)
   → covered by scalasca and vampir
- no suitable hardware usage (e.g. cache utilization)



## If CPU time is not enough ...

... getting hardware counter data from IBM HPC Toolkit:

- i/ hpccount command provides
  - execution wall clock time
  - resource utilization statistics
  - hardware performance counters information
  - derived hardware metrics

for the whole application run:

poe hpccount -u -n -o <name> <prog>



### hpccount options

- -g specifies the hardware counter group
- -n suppresses output to stdout
- -o writes output to file <name>
- -u unique file names will be used

HPM\_ASC\_OUTPUT, HPM\_VIZ\_OUTPUT for ASCII or XML output HPM\_AGGREGATE:

- mirror.so: gets raw data from each MPI-task [default]
- average.so: counter groups distributed in round robin fashion! Aggregator takes avg over these subgroups



### hpccount example

00263 75375 41806

99.763 %

hpccount (IBM HPC Toolkit v5.2.2.4) summary					
######Inscalaring #######ESOINCE Usage Statistics ######## 10km_60mm.per					
Total amount of time in system mode	: 0 : 4 : 7 : 3 : 6 : 0 : 0 : 0 : 0 : 0 : 0 : 0 : 0 : 0 : 0	0.0531 01968 7773 k 443961 925	i3446 seconds 128 seconds 1 kbytes kbytes*sec 194 Kbytes*se		
######################################					
Execution time (wall clock time) : 121.6	3412				
PM_FPU_1FLOP (FPU executed one flop instruc PM_FPU_FMA (FPU executed multiply-add instr PM_FPU_FSORT FDIV (FPU executed FSORT or FD			uction)		760 160000
PM_FPU_FLOP (FPU executed 1FLOP, FMA, FSORT PM_RUN_INST_CMPL (Run instructions complete PM_RUN_CYC (Run cycles)					160760 1519531 5648684
Utilization rate Instructions per run cycle Total scalar floating point operations			98.724 0.269		
Scalar flops / user time Algebraic flopating point operations			0.269 32076.000 263.709 267.116 32076.000	Mflop Mflop M	
Algebraic flop rate (flops / WCT) <sup>Z</sup> Algebraic flops / user time			263.709	мтіор	





### If CPU time is not enough ...

#### ... getting hardware counter data from IBM HPC Toolkit: ii/ instrumentation with libhpm

```
#include <libhpc.h>
int main(void) {
 MPI_Init(MPI_COMM_WORLD);
 MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  hpmInit(myrank, "my_program");
  ... maybe some initialization code
  hpmStart(1, "first section");
  ... some code you want to analyze
  hpmStop(1):
  ... more boring code
  hpmTerminate(myrank);
 MPI Finalize():
J,
```

```
PROGRAM HELLO_WORLD
                                              IMPLICIT NONE
                                            #include "f hpc.h"
                                              CALL MPI_INIT(ierror)
                                              CALL MPI COMM RANK(MPI COMM WORLD, mvrank, ierror)
                                              CALL f_hpminit(myrank, 'my_program')
                                              ... maybe some initialization code
                                              CALL f hpmstart(1, 'first section')
                                              ... some code you want to analyze
                                              CALL f_hpmstop(1)
                                              ... more boring code
                                              CALL f hpmterminate(mvrank)
                                              CALL MPT FINALIZE(ierror)
                                            FND
don't forget -I/usr/lpp/ppe.hpct/include and
-L/usr/lpp/ppe.hpct/lib64 -lhpc -lpmapi
```



## Choosing counters

on POWER6

- you can measure 6 counters simultaneously
- not all combinations allowed
- ▶ 202 performance counter groups (pmlist -g -1)
- may need to sample multiple times for completeness
- or use multiplexing of counter groups

Use hpccount/libhpc to measure code efficiency by means of:

- Instructions per run cycle, Mflop/s (group 127)
- L1 cache usage (group 47)
- cache/memory access (group 7,11)



#### counters are difficult to understand

- limited documentation
- experience needed to see counter value indicating a problem
- ▶ use IPC (instructions per run cycle) for first estimate

IPC = PM\_RUN\_INST\_CMPL / PM\_RUN\_CYC = 1/CPI

category	IPC	description
1	< 0.4	Houston, we have a problem
2	0.4 0.7	not tuned for POWER6
3	0.70.9	acceptable
4	0.91.3	very good (can be tough to get here)
5	> 1.3	wow ! (not always possible)
6	> 2.0	LINPACK, VMASS, ESSL, FFTW,



28/33

group 127:

- PM\_FPU\_1FLOP: FPU executed single FLOP operation
- PM\_FPU\_FMA: FPU executed a multiply-add
- PM\_FPU\_FSQRT\_FDIV
- ▶ PM\_FPU\_FLOP: 1flop, fma, sqrt, div operations for unit0 and 1

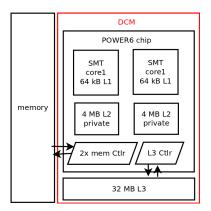
group 47:

- PM\_ST\_REF\_L1: L1 D cache store references
- PM\_LD\_REF\_L1: L1 D cache load references
- PM\_ST\_MISS\_L1: L1 D cache store misses
- PM\_LD\_MISS\_L1: L1 D cache load misses



group 7:

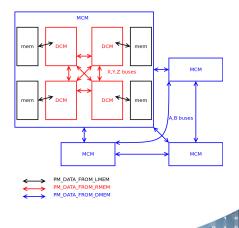
- PM\_DATA\_FROM\_L2: D Cache reloaded from local L2
- PM\_DATA\_FROM\_L21: D Cache reloaded from private L2 of other core on chip
- PM\_DATA\_FROM\_L2MISS: D Cache reloaded but not from local L2
- PM\_DATA\_FROM\_L3MISS: D Cache reloaded from beyond L3





#### group 11:

- PM\_DATA\_FROM\_LMEM: D Cache reloaded from local memory (attached to DCM)
- PM\_DATA\_FROM\_RMEM: D Cache reloaded from remote memory (attached to other DCM on same MCM)
- PM\_DATA\_FROM\_DMEM: D Cache reloaded from distant memory (attached to a different MCM)





### HPCC results out of the box

31/33

i/ HPL: solves a dense linear system in double precision (linpack)  $N = 20000, NB = 20 \lor 120, P = 8, Q = 4$ 

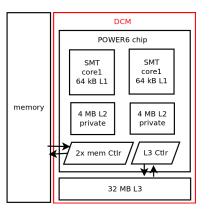
			libhpc: per MPI task						
NB	HPL Gflop/s	IPC		Gflop/s	% peak		L1 cache hit	rate	
20	2.39e+02	1.45 - 1	1.72	7.33 - 7.56	39.9 -	41.1 %	97.5 - 99.5	%	
120	3.15e+02	1.82 - 2.09		9.75 - 9.8	52.3 - 56.0 %		98.5 - 99.4	%	
		libhpc: per MPI task							
NB	PM_DATA_FR	DM_L2	PM_	DATA_FROM_L2	2MISS   PM_DATA_FROM_LMEM				
20	35131005 - 70	195962 8913581 - 12180			0711 5921680 - 9110732				
120	30105907 - 65	539464	49	992461 - 6045	666	333296	6 - 4227469		



## POWER6 background

POWER6 cache/memory: size and latency

- L1: 64 kB, 2 cycles
- L2: 4 MB core-private, 25 cycles
- L3: 32 MB chip-shared, 150 cycles
- mem: 50 (100) GB, 500 cycles





## HPCC results out of the box

ii/ DGEMM: measures floating point rate of double precision real matrix-matrix multiplication bench N = 1500 using libessl or not

			libhpc: per MPI task						
essl	HPL Gflop/s (r	HPL Gflop/s (min/max)			Gflop/s				
no	2.94 / 3.	0.65 - 0.68		3.4 - 3.6					
yes	10.84 / 1	2.49 - 2.59		14.2 - 14.8	l				
		libhpc: per MPI task							
essl	% peak	PM_ST_M	ISS_L1   PN		1_LD_MISS_L1				
no	18.5 - 19.6 %	1.23e6 -	1.47e6	6 215.68e6 - 221.9		e6			
yes	77.3 - 80.6 %	2.71e6 -	3.32e6 (		93e6 - 1.99e6				

