

GEM Model (Re)Vectorization

Prepared by: Ron McTaggart-Cowan

Date: 4 May 2006

Outline

- Model Version: 3.2.2
- Physics Version: 4.4

The purpose of this project is to re-implement vectorization in the GEM model, which has not been supported since the switch to scalar processors at the Meteorological Service. The vectorization and optimization tasks were undertaken using the NEC SX-6 'pharos2' at Ouranos, running the SUPER-UX operating system. With the exception of the bug-fix outlined in the O7 optimization, the results from the vector modified code bit matches (using a binary diff) with the original GEM version 3.2.2 release. All tests were made using 6 steps (of 10 min each for a totally of 1 h of simulation time) on a 250x50 domain, taking advantage of the FFT package on a grid of similar extent to that used in the LACES project. The majority of the optimization steps are very simple, and the speed-up for each is shown using the `-ftrace` SX-6 compiler option. This is a standard compiler option, and does not incur any significant overhead. Since the `-vopt` optimizing option is standard for compiling, this option was also used during the vectorization and optimization processes.

Summarized performance statistics are provided for each step of the optimization, along with notes about the changes and recommendations about each optimizing step. None of these vectorizing or optimizing modifications should have any significant impact on scalar run-times; however a wall-clock comparison of the fully modified and clean version 3.2.2 codes should be undertaken on the IBM system. The maintenance requirements of the modifications should therefore be minimal, since there is only minimal splitting of the code between implementations for vector and scalar architectures. As noted in the detailed sections, the vast majority of the full optimization can be obtained without any additional maintenance.

Basic Performance

The wallclock time and performance statistics for each of the GEM subprogram elements is derived from the traced output on the Ouranos NEC system. Only the main routines – from a wallclock consumption perspective – are included with this document, although full traces for all routines are available. This trace shows the out-of-the-box performance of the clean GEM model version 3.2.2.

PROG.UNIT	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONF
adw_tricub_lag3d	72	38.906(46.5)	540.367	460.3	193.0	0.00	0.0	0.000	0.0006	2.8449	0.0000
kfcp4	294	7.513(9.0)	25.556	594.9	134.7	52.15	152.6	0.631	0.0096	0.9000	0.0001
adw_ckbd_lam\$1	88	5.594(6.7)	63.569	628.3	33.0	0.00	112.0	0.000	0.0005	0.4987	0.0000

consun1	294	4.391(5.2)	14.937	1674.8	529.2	78.13	246.2	0.662	0.0047	0.5594	0.0001
radir8	126	2.878(3.4)	22.844	3567.9	1082.4	99.35	246.1	2.834	0.0014	0.0004	0.0901
qcfft8	4560	2.723(3.3)	0.597	1066.6	421.9	72.07	83.8	0.521	0.0540	0.8637	0.0147
adw_trilin_turbo\$1	114	2.367(2.8)	20.766	4737.7	839.9	99.85	248.2	2.363	0.0002	0.0005	0.0467
consun1.fmroft	3399228	2.147(2.6)	0.001	459.3	63.5	0.00	0.0	0.000	0.0002	0.1322	0.0000
adw_trajsp\$1	38	1.492(1.8)	39.254	7876.4	3042.3	99.50	250.9	1.445	0.0003	0.0010	0.0000
agregel	294	1.281(1.5)	4.357	451.8	59.6	13.28	247.1	0.025	0.0023	0.0127	0.0003
total	4123008	83.712(100.0)	0.020	1657.0	541.6	78.38	206.4	18.112	0.4066	6.2456	0.1977

Bounds Check and Interpolation (O1)

The O1 optimizations are very basic in nature and are responsible for the majority of the enhancements made during the vectorization / optimization process. In the bounds checking routine for the LAM version of the model (`adw_ckbd_lam.ftn`), print statements in the loops prevented vectorization, so a set of flags were implemented instead. The functionality of the routine remains identical to that of the original version. The 3D interpolation routine (`adw_tricub_lag3d.ftn`) has a set of conditional statements that define vectors in only a single branch. This construct prevents vectorization and was modified to include null definitions in the opposite case. Although this appears to have no effect on the results since the (local) initialized values are never referenced in the routine, M. Valin expressed some concern from a memory perspective because of the branch guessing capability of the SX-6. The O1 optimizations are very high priority and are readily implemented.

PROG.UNIT	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONF
kfcp4	294	7.065(15.9)	24.029	638.4	147.4	53.34	155.2	0.639	0.0101	0.5833	0.0001
adw_tricub_lag3d	72	6.048(13.6)	84.003	5561.2	1285.2	99.95	244.1	6.048	0.0002	0.0004	0.1384
consun1	294	4.878(11.0)	16.593	1483.6	468.6	78.13	242.2	0.668	0.0058	1.0390	0.0001
hzd_solmxma\$1	24	3.865(8.7)	161.061	5256.5	3178.4	96.91	45.3	3.590	0.0006	0.1699	0.0000
sol_mxma8_2\$1	24	3.025(6.8)	126.055	4871.4	3130.4	97.71	37.0	2.735	0.0005	0.1794	0.0000
adw_trilin_turbo\$1	114	2.376(5.4)	20.838	4647.6	823.5	99.85	244.4	2.371	0.0002	0.0005	0.0398
consun1.fmroft	3343956	2.105(4.7)	0.001	460.8	63.7	0.00	0.0	0.000	0.0005	0.1054	0.0000
radir7	84	1.897(4.3)	22.583	3551.2	1077.1	99.34	242.1	1.867	0.0011	0.0004	0.0594
adw_trajsp\$1	38	1.525(3.4)	40.136	7581.4	2927.4	99.49	249.8	1.478	0.0003	0.0028	0.0000
vco2inf2	84	0.798(1.8)	9.501	6184.2	1958.2	98.47	242.5	0.709	0.0005	0.0364	0.0001
total	3461542	44.339(100.0)	0.013	3946.8	1441.1	95.94	119.1	28.475	0.2600	2.4583	0.2809

Horizontal Diffusion (O2)

The rationale for this optimization focuses on the small vector length noted in the `hzd_solmxma.ftn` routine. Reshaping the arrays in this subprogram should allow for an increase in the mean vector length; however, its wallclock timing did not change significantly. Since these modifications were did not enhance the vector length, op rate or wallclock timing, their inclusion is entirely optional.

PROG.UNIT	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONF
kfcp4	294	7.067(16.0)	24.038	638.2	147.3	53.34	155.2	0.642	0.0186	0.5968	0.0001
adw_tricub_lag3d	72	6.070(13.7)	84.300	5541.6	1280.7	99.95	244.1	6.068	0.0025	0.0033	0.1383
consun1	294	4.903(11.1)	16.677	1476.1	466.3	78.13	242.2	0.672	0.0164	1.0591	0.0001

hzd_solmxma\$1	24	3.833(8.7)	159.717	5300.8	3205.1	96.91	45.3	3.564	0.0038	0.1630	0.0000
sol_mxma8_2\$1	24	3.016(6.8)	125.687	4885.7	3139.6	97.71	37.0	2.733	0.0029	0.1727	0.0000
adw_trilin_turbo\$1	114	2.380(5.4)	20.875	4639.4	822.0	99.85	244.4	2.375	0.0012	0.0017	0.0398
consun1_fmroft	3343956	2.101(4.7)	0.001	461.5	63.8	0.00	0.0	0.000	0.0048	0.1088	0.0000
radir7	84	1.901(4.3)	22.632	3543.5	1074.8	99.34	242.1	1.871	0.0011	0.0003	0.0593
adw_trajsp\$1	38	1.522(3.4)	40.053	7597.1	2933.5	99.49	249.8	1.473	0.0019	0.0037	0.0000
vco2inf2	84	0.814(1.8)	9.685	6066.8	1921.0	98.47	242.5	0.724	0.0005	0.0368	0.0001
total	3461370	44.304(100.0)	0.013	3917.9	1430.3	95.92	118.6	28.422	0.3163	2.5011	0.2796

Advection Interpolators (O3)

A set of routines is modified for this optimization (`adw_interp2.ftn`, `adw_main_3_intlag.ftn` and `adw_comp.cdk`). Additionally, a new routine is added (`adw_tricub_lag3d_vec.ftn`), the only vector-specific branch in the modified code. The main improvement in O3 was seen in the common namespace storage of array indices, rather than a series of re-computations throughout the interpolator code. If this strategy is found to be effective on scalar machines, then the preprocessor branch isolating vector/scalar architectures in `adw_interp2.ftn` could be removed and `adw_tricub_lag3d_vec.ftn` used to replace the original `adw_tricub_lag3d_vec.ftn`. This would simplify maintenance and reduce code redundancy. The increased vector length may justify the inclusion of this set of modifications, however the overall flop rate seems to have dropped a bit.

PROG.UNIT	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONF
kfcp4	294	7.502(16.8)	25.516	595.8	134.9	52.15	152.6	0.631	0.0123	0.8854	0.0001
adw_tricub_lag3d	72	6.029(13.5)	83.734	5669.6	1310.5	99.95	248.1	6.028	0.0008	0.0009	0.1698
consun1	294	4.185(9.4)	14.235	1757.4	555.3	78.13	246.2	0.664	0.0091	0.2985	0.0001
radir8	126	2.882(6.5)	22.874	3563.2	1081.0	99.35	246.1	2.838	0.0017	0.0008	0.0901
qcfft8	4560	2.735(6.1)	0.600	1062.1	420.1	72.07	83.8	0.525	0.0494	0.8763	0.0149
adw_trilin_turbo\$1	114	2.360(5.3)	20.702	4752.4	842.5	99.85	248.2	2.355	0.0007	0.0007	0.0467
consun1_fmroft	3399228	2.061(4.6)	0.001	478.3	66.1	0.00	0.0	0.000	0.0013	0.0272	0.0000
adw_trajsp\$1	38	1.492(3.3)	39.275	7872.2	3040.7	99.50	250.9	1.445	0.0009	0.0013	0.0000
vco2inf2	126	1.126(2.5)	8.933	6794.0	2117.0	98.71	246.3	1.083	0.0008	0.0004	0.0002
hzd_solfft_lam\$1	24	0.888(2.0)	36.994	2952.3	718.1	88.88	61.0	0.866	0.0027	0.0040	0.0001
total	4123396	44.561(100.0)	0.011	3451.6	1038.1	94.69	215.6	24.736	0.4314	2.6117	0.3733

Radiation Scheme (O4)

All indirect references in the radiation scheme (`radir7.ftn`) that occur more than once in a loop are replaced with scalar temporaries. This leads to the full optimization of the subprogram, but does not appear to improve either the wallclock or op-based performance indicators. As a result, the O4 optimization appears to be entirely optional, unless such a referencing scheme affects performance on scalar machines.

PROG.UNIT	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONF
kfcp4	294	6.934(18.4)	23.585	650.4	150.2	53.34	155.2	0.646	0.0094	0.4278	0.0001
adw_tricub_lag3d_vec	72	4.403(11.7)	61.156	6992.3	1702.1	99.87	256.0	4.397	0.0008	0.0008	0.0950

adw_trajjsp\$1	114	2.382(7.1)	20.893	4708.8	834.8	99.85	248.2	2.377	0.0002	0.0005	0.0467
agregel	38	1.509(4.5)	39.711	7785.7	3007.2	99.50	250.9	1.462	0.0004	0.0016	0.0000
vco2inf2	294	1.277(3.8)	4.342	453.3	59.9	13.28	247.1	0.025	0.0024	0.0083	0.0003
hzd_solfft_lam\$1	126	1.122(3.4)	8.905	6815.5	2123.8	98.71	246.3	1.079	0.0007	0.0002	0.0002
consun1	24	0.876(2.6)	36.517	2990.9	727.4	88.88	61.0	0.857	0.0012	0.0034	0.0001
flxsurf3.fmi	294	0.767(2.3)	2.608	8614.2	3077.0	99.38	246.1	0.738	0.0042	0.0048	0.0001
	217431	0.743(2.2)	0.003	311.4	79.7	0.00	0.0	0.000	0.0015	0.0735	0.0000
total	724227	33.471(100.0)	0.046	4522.8	1382.4	97.34	217.5	24.834	0.3462	1.4297	0.3716

Surface Module Summation (O7)

The implementation of the urban land use scheme appears to have introduced a memory bug for the SX-6 platform. This is one of the five surface modules called by the aggregating routine `agregel.ftn`, and was not accessed during the vectorization and optimization procedures; however, the removal of branches referencing the urban scheme yields a different result from their evaluation as false conditionals. This is clearly incorrect, but the source of the problem is not obvious. Extensive notes outline the problem have been added to the `agregel.ftn` routine, and an preprocessor directive results in a fatal warning if a NEC user tries to access the urban land use scheme. The functionality of this code has been verified as identical to the unmodified version on scalar platforms, and given that the magnitude of the errors on the vector machine are not small (up to 20% in the maximum vertical motion after 1 h of simulation time), at least some form of this fix/warning should be included to alert users to possible problems with the scheme.

PROG.UNIT	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONF
adw_tricub_lag3d	72	5.976(19.0)	83.002	5719.5	1322.0	99.95	248.1	5.976	0.0002	0.0002	0.1698
radir8	126	2.852(9.1)	22.636	3600.7	1092.4	99.35	246.1	2.808	0.0014	0.0005	0.0899
qcfft8	4560	2.690(8.5)	0.590	1079.9	427.2	72.07	83.8	0.512	0.0497	0.8471	0.0141
adw_trilin_turbo\$1	114	2.351(7.5)	20.624	4770.4	845.7	99.85	248.2	2.346	0.0002	0.0004	0.0467
adw_trajjsp\$1	38	1.485(4.7)	39.076	7912.3	3056.1	99.50	250.9	1.439	0.0003	0.0009	0.0000
vco2inf2	126	1.106(3.5)	8.779	6913.7	2154.3	98.71	246.3	1.063	0.0006	0.0002	0.0003
hzd_solfft_lam\$1	24	0.846(2.7)	35.238	3099.5	753.8	88.88	61.0	0.826	0.0012	0.0032	0.0001
consun1	294	0.754(2.4)	2.563	8763.1	3130.2	99.38	246.1	0.726	0.0035	0.0044	0.0001
flxsurf3.fmi	217431	0.681(2.2)	0.003	339.8	87.0	0.00	0.0	0.000	0.0014	0.0030	0.0000
flxsurf3.fhi	217431	0.490(1.6)	0.002	312.8	72.8	0.00	0.0	0.000	0.0007	0.0045	0.0000
total	724168	31.498(100.0)	0.043	4781.9	1463.5	97.66	217.4	24.311	0.3357	1.2665	0.3698

FFT Array Referencing (O8)

The modifications to the FFT preparation routine (`qcfft8.ftn`) are extensive, and focus on the restructuring of references to both argument and internal arrays. In the original code, 2D arrays are referenced using 1D vector pointers in the subprogram, with complicated indirect referencing computed using a statement function. In the modified code, 2D arrays remain 2D arrays, and references are direct, although irregular strides will likely reduce the effectiveness of vectorization. Although the individual performance enhancement is relatively small for each time this subprogram is called, the frequency with which it is called makes these modifications relatively important. In addition, the 2D referencing dramatically improves code readability since the gridded location of each

element is easily determined.

PROG.UNIT	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONF
adw_tricub_lag3d	72	6.037(19.3)	83.850	5661.7	1308.7	99.95	248.1	6.037	0.0002	0.0003	0.1699
radir8	126	2.867(9.2)	22.754	3582.1	1086.7	99.35	246.1	2.823	0.0014	0.0005	0.0902
adw_trilin_turbo\$1	114	2.360(7.5)	20.702	4752.4	842.5	99.85	248.2	2.355	0.0003	0.0005	0.0467
qcfft8	4560	2.255(7.2)	0.494	1475.6	519.1	81.37	129.8	0.537	0.0552	0.6688	0.0194
adw_trajsp\$1	38	1.492(4.8)	39.272	7872.7	3040.9	99.50	250.9	1.446	0.0005	0.0011	0.0000
vco2inf2	126	1.109(3.5)	8.802	6894.9	2148.5	98.71	246.3	1.066	0.0007	0.0003	0.0002
hzd_solfft_lam\$1	24	0.851(2.7)	35.439	3081.8	749.6	88.88	61.0	0.831	0.0018	0.0032	0.0000
consun1	294	0.758(2.4)	2.578	8713.3	3112.4	99.38	246.1	0.730	0.0036	0.0045	0.0001
flxsurf3.fmi	217431	0.753(2.4)	0.003	307.2	78.7	0.00	0.0	0.000	0.0016	0.1016	0.0000
flxsurf3.fhi	217431	0.491(1.6)	0.002	312.3	72.6	0.00	0.0	0.000	0.0007	0.0050	0.0000
total	724168	31.304(100.0)	0.043	4825.1	1473.3	97.80	219.7	24.487	0.3455	1.1958	0.3776

PBL Stability Calculation (O9)

The stability of the boundary layer is used to compute surface fluxes in several subprograms, including `diasurf2.ftn`, `flxsurf3.ftn`, `slfun_tq.ftn`, and `slfun_uv.ftn`. Functions for calculating various stability-based parameters are found in `stabfunc.cdk`. These functions are called in loops from the flux subprograms and inhibit vectorization. The usual solution for this is to inline the (short) function using a compiler directive and the `-pi` compiler option. However, these functions access the namespace of the calling program directly, a bizarre behaviour that prevents compile-time inlining. The interfaces of the `stabfunc.cdk` functions have been modified to pass all array variables required in the functions themselves, and compiler directives have been added to ensure inlining. This leads to full vectorization of the loops that contain the function calls, and a significant overall speedup.

It seems to me that some more extensive recoding than simple vectorization and optimization is required to correct problems with these calculations. As far as I know, all code was supposed to have been removed from “comdec” files. This is clearly not the case for `stabfunc.cdk`, a comdec that contains three separate non-trivial functions. This comdec should be replaced with a set of separate functions or a module. As currently written, the `stabfunc.cdk` code and all subprograms that call functions within it are essentially un-maintainable. Because the functions inherit the namespace of the parent program (both scalar and array variables from the calling program are referenced in the comdec’s functions without passing through the interface argument list), the naming scheme in the calling programs and the comdec must be identical. Clearly, this is all a complete violation of modular programming and virtually ensures that the collective code is essentially unmanageable. Furthermore, undeclared variables exist in the comdec subprograms that must be declared by the parent, even if they are not defined or referenced in the parent’s code (see the “X” variable in `slfun_tq.ftn` for example). Since there is no documentation in the comdec, it is virtually impossible for a developer to know what these variables should be, and what values they should contain (if any). And it gets worse; the functions actually make use of their access to the parent’s namespace to modify values within it.

For example, the “X” variable is declared in `flxsurf3.ftn`, defined internally in calls to the `comdec` functions (although it is not passed through the interface or returned), and then referenced later in `flxsurf3.ftn`. This is a horrendous violation of privacy and variable scope that appears to be completely unnecessary. During the vectorization and optimization process, I fixed only the aspect of this programming nightmare that were required to satisfy the optimizing compiler; however, this gross violation of basic programming principles should be addressed as soon as possible for the sake of modularity and maintainability.

PROG.UNIT	FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	I-CACHE MISS	O-CACHE MISS	BANK CONF
adw_tricub_lag3d	72	6.074(20.7)	84.355	5627.8	1300.8	99.95	248.1	6.073	0.0001	0.0003	0.1699
radir8	126	2.867(9.8)	22.756	3581.7	1086.6	99.35	246.1	2.823	0.0015	0.0005	0.0904
adw_trilin_turbo\$1	114	2.371(8.1)	20.794	4731.3	838.7	99.85	248.2	2.366	0.0003	0.0004	0.0467
qcfft8	4560	2.275(7.8)	0.499	1462.8	514.6	81.37	129.8	0.544	0.0564	0.6780	0.0195
adw_trajsp\$1	38	1.500(5.1)	39.486	7830.1	3024.4	99.50	250.9	1.454	0.0003	0.0010	0.0000
vco2inf2	126	1.109(3.8)	8.801	6896.3	2148.9	98.71	246.3	1.066	0.0007	0.0003	0.0002
hzd_solfft_lam\$1	24	0.874(3.0)	36.425	2998.4	729.3	88.88	61.0	0.855	0.0015	0.0034	0.0001
consun1	294	0.763(2.6)	2.596	8652.6	3090.7	99.38	246.1	0.735	0.0039	0.0045	0.0001
wflux	17766	0.485(1.7)	0.027	4967.6	1919.1	97.61	245.7	0.265	0.0272	0.0607	0.0001
kfcfp4	294	0.475(1.6)	1.617	6361.0	2009.1	98.60	229.6	0.411	0.0078	0.0073	0.0006
total	130896	29.313(100.0)	0.224	5139.6	1572.2	98.26	219.5	24.696	0.3413	1.0536	0.3774