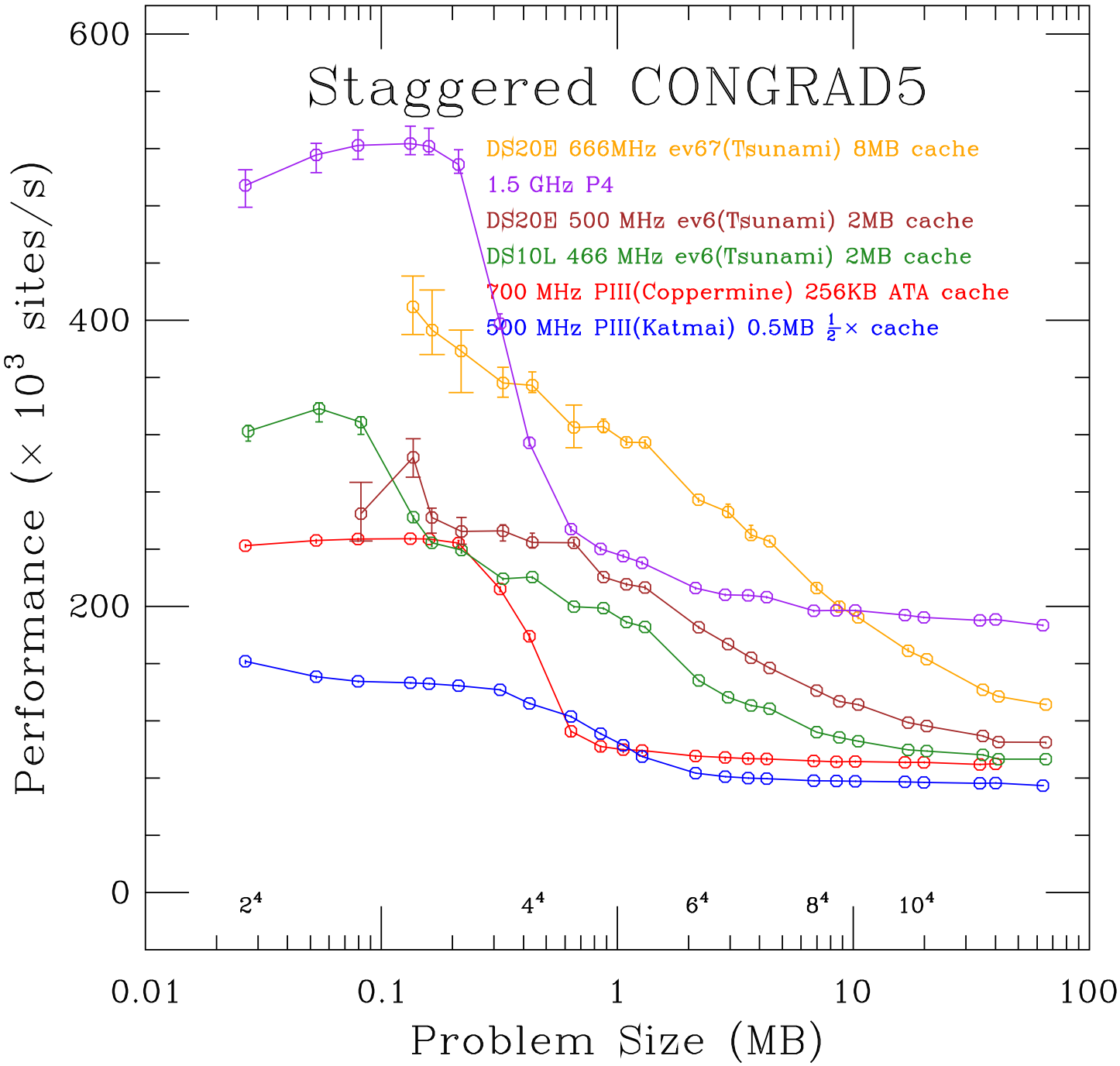

Staggered CONGRAD5
DS20E 666MHz ev67(Tsunami) 8MB cache
1.5 GHz P4
DS20E 500 MHz ev6(Tsunami) 2MB cache
DS10L 466 MHz ev6(Tsunami) 2MB cache
700 MHz PIII(Coppermine) 256KB ATA cache
500 MHz PIII(Katmai) 0.5MB $\frac{1}{2}\times$ cache
Performance ($\times 10^3$ sites/s)
600
400
200
0
2^4
4^4
6^4
8^4
10^4
0.01
0.1
1
10
100
Problem Size (MB)

x86 Optimization via SSE

- SSE = Streaming SIMD Extensions
- SIMD
  - Uses an 8 register, 128-bit wide stack
  - Separate from MMX stack
  - Unlike MMX, not shared with FPU stack
  - Allows 4-wide simple single precision Flop's
- Streaming
  - Prefetch hints
  - Non-temporal - to L1 cache only
  - Temporal - to all levels of cache
- GCC support
  - None, that I know of
  - We implement prefetch instructions via macros
  - We use NASM assembler for SIMD codes
- Linux support
  - Requires additional code for proper context switches
  - For 2.2.x kernel, we use PIII patch set from Andrea Arcangeli
  - Unsure of 2.4.x kernel support
  - Although there used to be gdb support, it's no longer current

Pentium III (Coppermine) Timings

	MILC	SSE
mult_su3_mat_vec	131	102
mult_adj_su3_mat_vec	128	98
mult_su3_mat_vec_sum_4dir	599	369
mult_adj_su3_mat_vec_4dir	500	304
mult_adj_su3_mat_hwvec	297	178
mult_su3_mat_hwvec	297	184
mult_su3_nn	421 (851)	257
mult_su3_na	400	255
scalar_mult_add_su3_matrix	142	64
su3_projector	222	107

- Timings in machine cycles (700 MHz Pentium III, 100 MHz FSB)
- These numbers include subroutine and loop overheads of 14 cycles
- Sanity check:
  - FPU Multiply: 5 cycles
  - FPU Add: 3 cycles
  - SSE Multiply (4-wide): 4 cycles
  - SSE adds (4-wide): 3 cycles
  - su3 matrix X su3 matrix: 108 multiplies, 90 adds
    - 810 FPU cycles
    - 175 SSE multiply/add cycles

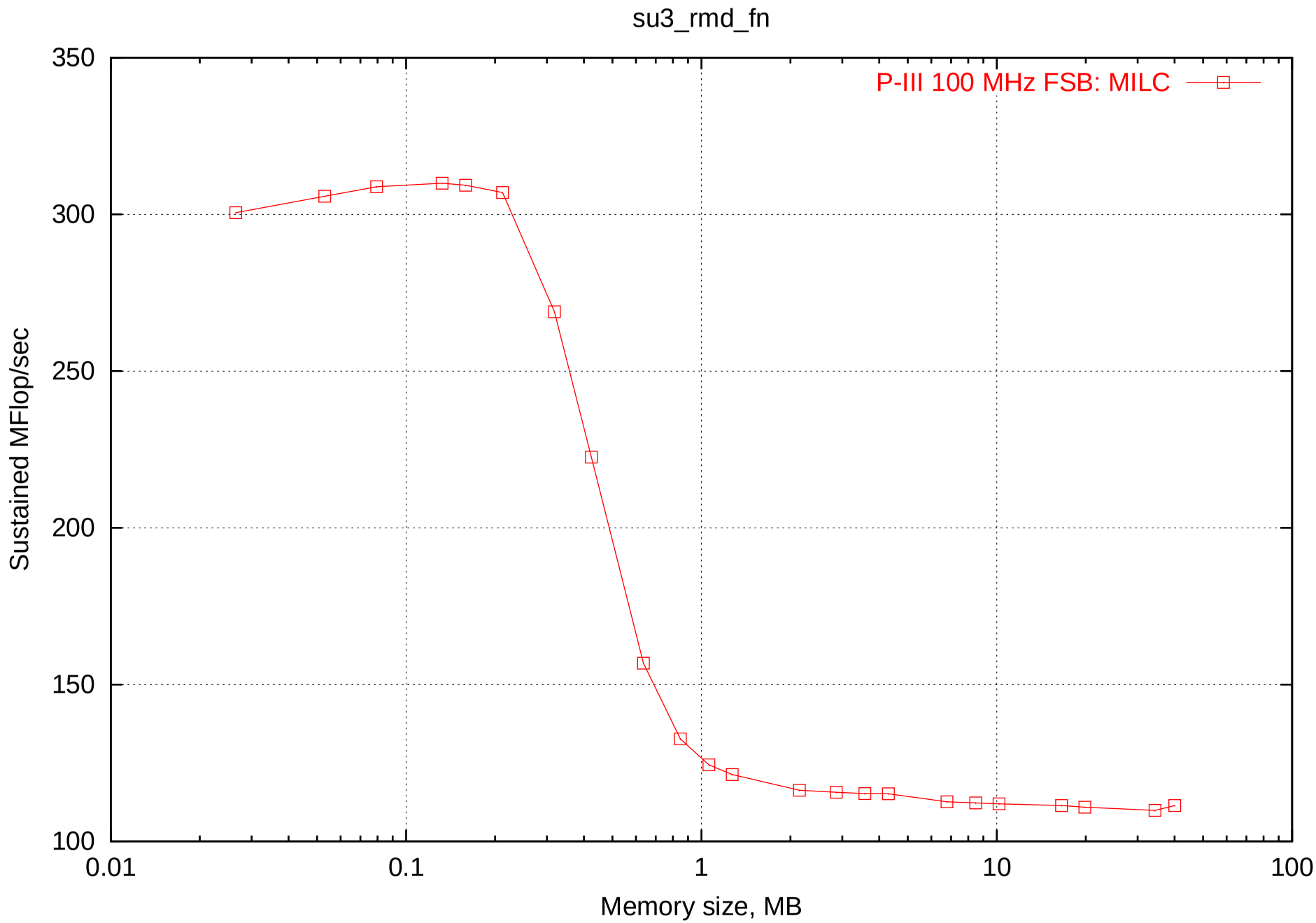
Pentium III (Coppermine) and Pentium 4 Timings

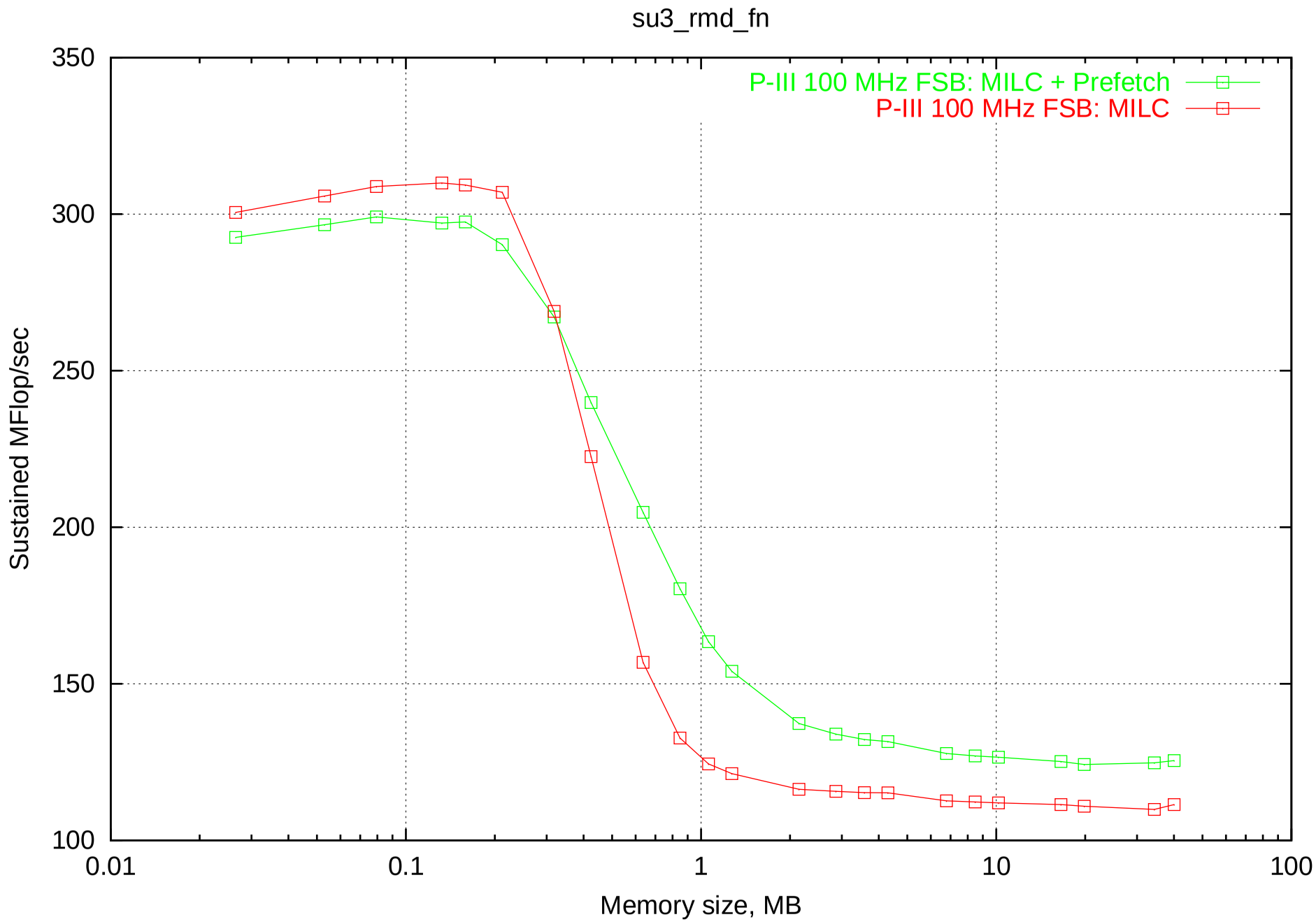
	P-III		P4	
	MILC	SSE	MILC	SSE
mult_su3_mat_vec	131	102	162	96
mult_adj_su3_mat_vec	128	98	116	101
mult_su3_mat_vec_sum_4dir	599	369	603	368
mult_adj_su3_mat_vec_4dir	500	304	548	362
mult_adj_su3_mat_hwvec	297	178	276	185
mult_su3_mat_hwvec	297	184	276	176
mult_su3_nn	421	257	399	251
mult_su3_na	400	255	386	253
scalar_mult_add_su3_matrix	142	64	152	74
su3_projector	222	107	234	102

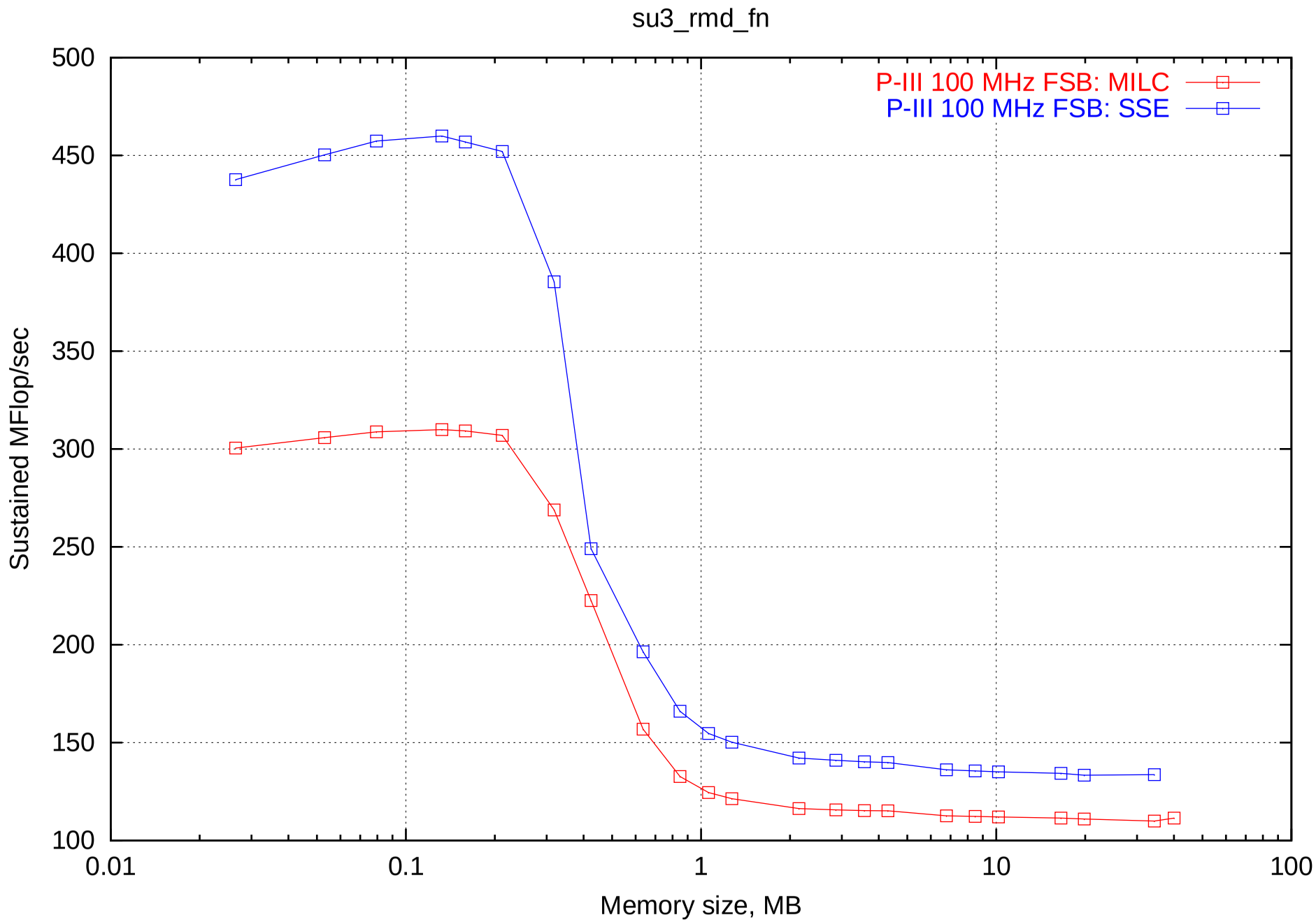
Prefetch Code Sample

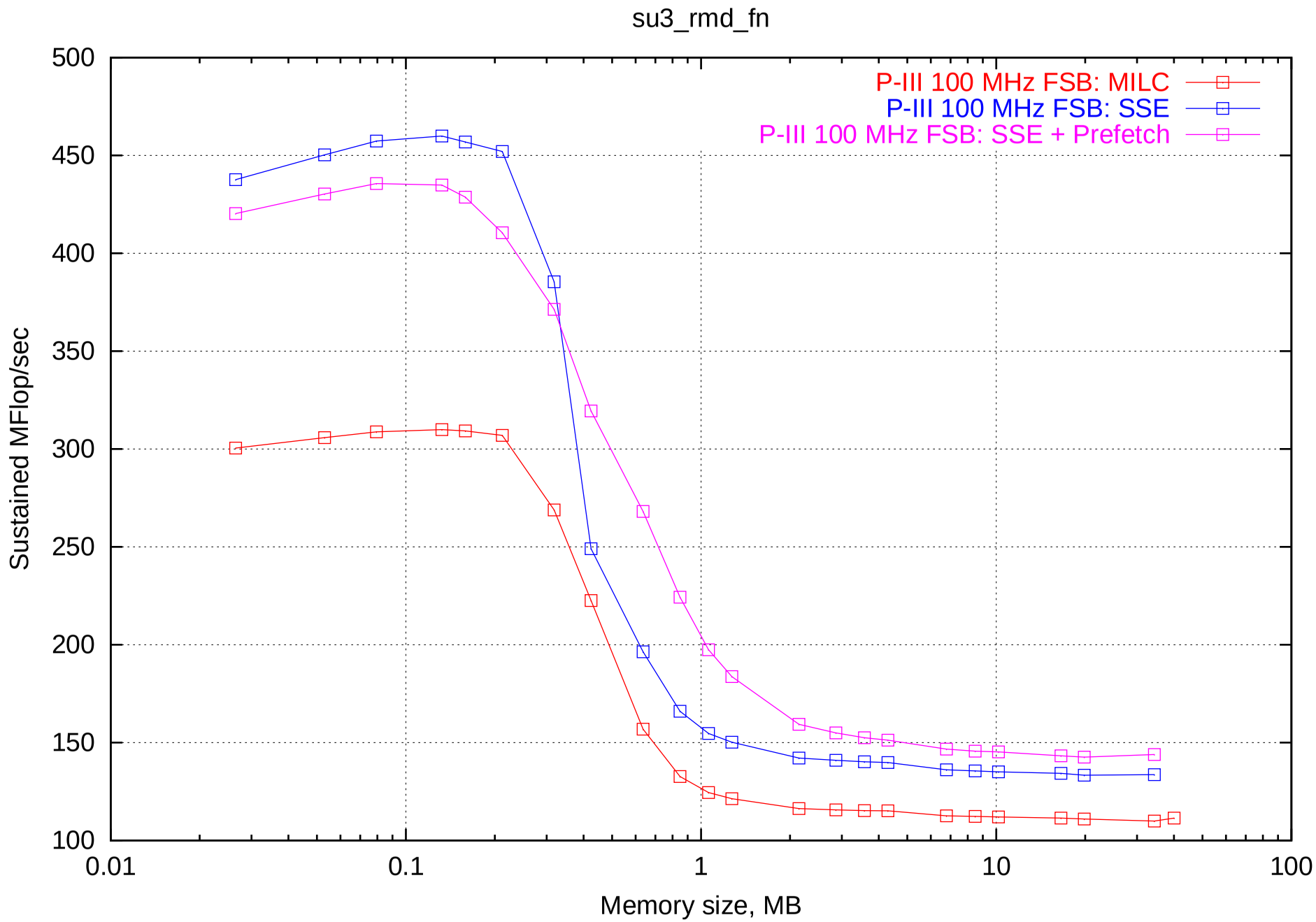
```
/* Use fat link for single link transport */
FORSOMEPARITY( i, s, otherparity ){
   PREFETCH( &(s->fatlink[0].e[0][0].real) );
   PREFETCH( &(s->fatlink[0].e[3][3].imag) );
   PREFETCH( &(s->fatlink[1].e[0][0].real) );
   PREFETCH( &(s->fatlink[1].e[3][3].imag) );
   PREFETCH( &(s->fatlink[2].e[0][0].real) );
   PREFETCH( &(s->fatlink[2].e[3][3].imag) );
   PREFETCH( &(s->fatlink[3].e[0][0].real) );
   PREFETCH( &(s->fatlink[3].e[3][3].imag) );
   PREFETCH( &(((su3 vector *)F PT(s,src))->c[0].real) );
   PREFETCH( &(((su3 vector *)F PT(s,src))->c[3].imag) );
   mult_adj_su3_mat_vec_4dir( s->fatlink,
    _   (su3 vector *)F PT(s,src), s->tempvec );
   /* multiply by 3-link matrices too */
   PREFETCH( &(s->longlink[0].e[0][0].real) );
   PREFETCH( &(s->longlink[0].e[3][3].imag) );
   PREFETCH( &(s->longlink[1].e[0][0].real) );
   PREFETCH( &(s->longlink[1].e[3][3].imag) );
   PREFETCH( &(s->longlink[2].e[0][0].real) );
   PREFETCH( &(s->longlink[2].e[3][3].imag) );
   PREFETCH( &(s->longlink[3].e[0][0].real) );
   PREFETCH( &(s->longlink[3].e[3][3].imag) );
   PREFETCH( &(((su3 vector *)F PT(s,src))->c[0].real) );
   PREFETCH( &(((su3 vector *)F PT(s,src))->c[3].imag) );
   mult_adj_su3_mat_vec_4dir( s->longlink,
    _   (su3 vector *)F PT(s,src), s->templongvec );
} END_LOOP
```

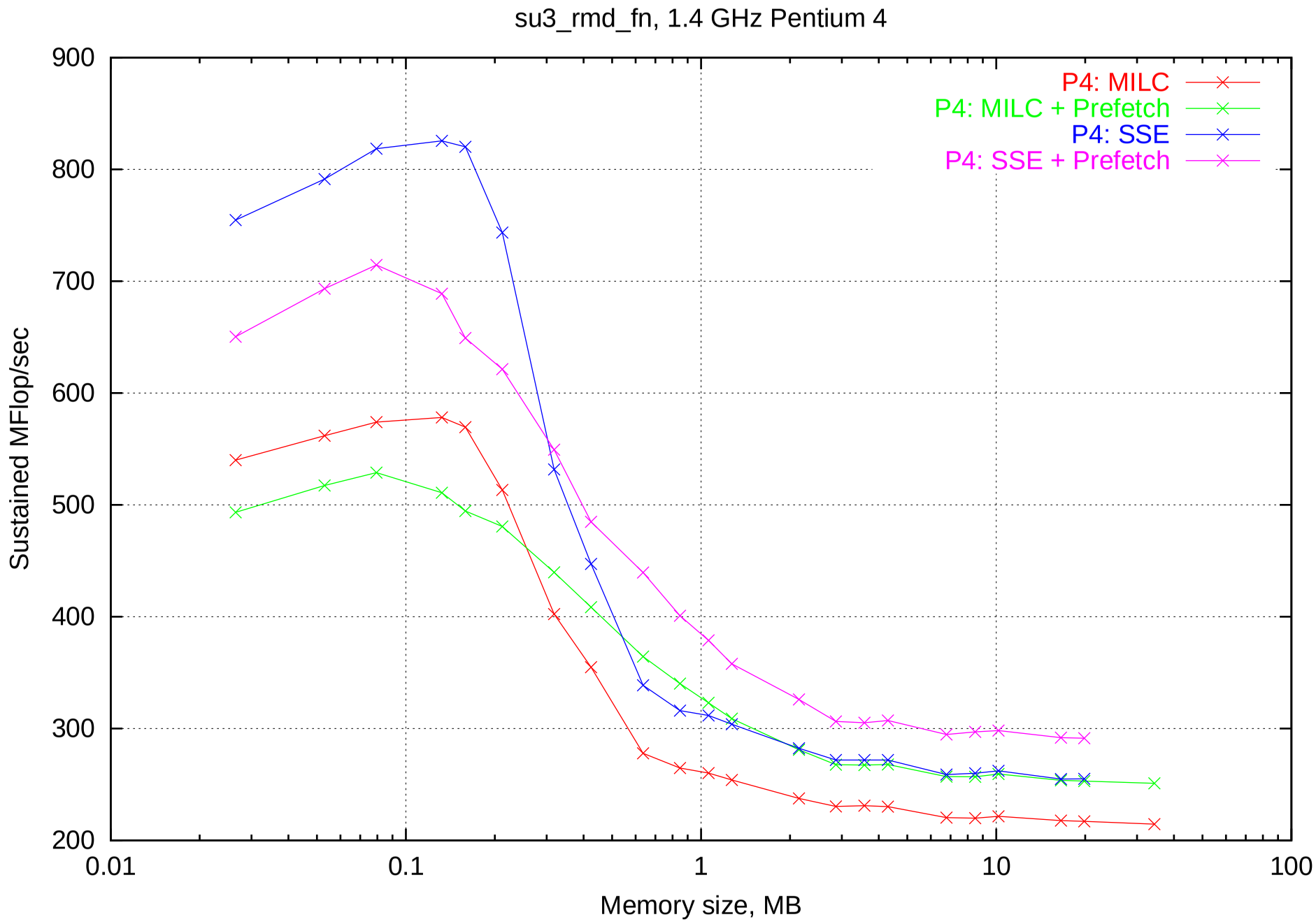
```
/****************** ks_imp_includes.h ******************************/
/*
*  Include files for Kogut-Susskind dynamical improved action application
*/

.
.
.

#define SSE
#ifdef SSE
#define mult_su3_mat_vec(args...) sse_mult_su3_mat_vec(## args)
#define mult_su3_mat_vec_sum_4dir(args...) sse_mult_su3_mat_vec_sum_4dir(## args)
#define mult_adj_su3_mat_vec(args...) sse_mult_adj_su3_mat_vec(## args)
#define mult_adj_su3_mat_vec_4dir(args...) sse_mult_adj_su3_mat_vec_4dir(## args)
#define mult_su3_mat_hwvec(args...) sse_mult_su3_mat_hwvec(## args)
#define mult_adj_su3_mat_hwvec(args...) sse_mult_adj_su3_mat_hwvec(## args)
#define mult_su3_nn(args...) sse_mult_su3_nn(## args)
#define mult_su3_na(args...) sse_mult_su3_na(## args)
#define su3_projector(args...) sse_su3_projector(## args)
#define scalar_mult_add_su3_matrix(args...) sse_scalar_mult_add_su3_matrix(## args)
#endif

#define USE_PREFETCH
#ifdef USE_PREFETCH
#   define PREFETCH_T0(  addr ) \
        __asm__( ".byte 0x0f,0x18,0x08" ::"a"(addr) )
#   define PREFETCH_T1(  addr ) \
        __asm__( ".byte 0x0f,0x18,0x10" ::"a"(addr) )
#   define PREFETCH_NTA( addr ) \
        __asm__( ".byte 0x0f,0x18,0x00" ::"a"(addr) )
#   define PREFETCH( addr ) \
        __asm__( ".byte 0x0f,0x18,0x00" ::"a"(addr) )
#else
#   define PREFETCH_T0( addr )
#   define PREFETCH_T1( addr )
#   define PREFETCH_NTA( addr )
#   define PREFETCH( addr )
#endif

.
.
.
```


su3_rmd_fn
P-III 100 MHz FSB: MILC
Sustained MFlop/sec
Memory size, MB
350
300
250
200
150
100
0.01
0.1
1
10
100


su3_rmd_fn
P-III 100 MHz FSB: MILC + Prefetch
P-III 100 MHz FSB: MILC
Sustained MFlop/sec
Memory size, MB
350
300
250
200
150
100
0.01
0.1
1
10
100


su3_rmd_fn
P-III 100 MHz FSB: MILC
P-III 100 MHz FSB: SSE
Sustained MFlop/sec
Memory size, MB
500
450
400
350
300
250
200
150
100
0.01
0.1
1
10
100


su3_rmd_fn
P-III 100 MHz FSB: MILC
P-III 100 MHz FSB: SSE
P-III 100 MHz FSB: SSE + Prefetch
Sustained MFlop/sec
Memory size, MB
500
450
400
350
300
250
200
150
100
0.01
0.1
1
10
100


su3_rmd_fn, 1.4 GHz Pentium 4
P4: MILC
P4: MILC + Prefetch
P4: SSE
P4: SSE + Prefetch
Sustained MFlop/sec
Memory size, MB
900
800
700
600
500
400
300
200
0.01
0.1
1
10
100