

Developing an Abstraction for Accelerator Programming

David Kunzman

```

    ori    $28, $0, $22, $26, 0
    fmov  $13, 0
    fmov  $15, $2, $48
    fmov  $28, $5, 0
    fmov  $12, 0
    fmov  $6, $2, $22
    fmov  $27, $10
    fmov  $1, 0
    fmov  $37, $32, $8
    lq    $2, $13, $48
    fmov  $36, $36, $21
    fmov  $72, $14
    fmov  $78, $43, $15
    hrr   $92, $90
    fmov  $12, $16, $7
    fmov  $11, $42, $15
    fmov  $23, $14, $72
    fmov  $79, $41, $15
    fmov  $5, $2, $8
    fmov  $35, $35, $25
    hrrp  $2
    ai    $20, $20, 1
    fmov  $16, $12, $19
    fmov  $19, $31, $24
    fmov  $81, $12, $17
    fmov  $17, $30, $22
    fmov  $51, $10, $48
    fmov  $22, $12, $18
    lq    $10, $13, $39
    fmov  $14, $6, $23
    fmov  $23, $22, 0
    fmov  $18, $27, $28
    fmov  $37, $19, $19
    fmov  $97, $10, $21
    fmov  $26, $44, $14
    fmov  $21, $17, $17, $3
    fmov  $97, $13, $39
    fmov  $8, $8, $26
    lq    $28, $13, $38
    fmov  $14, $18, $18, $21
    fmov  $21, $16, $26
    fmov  $24, $28, $25
    fmov  $25, $22, $26
    nop   $27
    frequent $4, $14
    fi    $77, $14, $4
    fmov  $24, $13, $38
    ai    $13, $13, 16
    lqd   $22, 0($15)
    ai    $15, $15, 16
    lq    $26, $33, $79
    lq    $24, $33, $78
    lq    $28, $33, $11
    fmov  $7, $7
    fi    $16, $7, $7
    l92:
    hrr   $28, $198
    l80:
    ori    $23, $22, 0
    fi    $15, $32, $23
    fmov  $20, $16
    fs    $11, $30, $22
    fmov  $12, $14
    fs    $22, $31, $24
    lqd   $5, 0($50)
    ai    $53, $53, 1
    cux  $23, $34, $42
    fi    $13, $16, $20
    cux  $31, $34, $46
    fi    $78, $14, $12
    cux  $30, $34, $45
    fi    $7, $17, $13

```

Intro

- This talk will focus on Cell processor
 - Cell Broadband Engine Architecture (CBEA)
 - Power Processing Element (PPE)
 - Synergistic Processing Element (SPE)
 - Current implementations
 - Sony Playstation 3 (1 chip with 6 SPEs)
 - IBM Blades (2 chips with 8 SPEs each)
 - Toshiba SpursEngine (1 chip with 4 SPES)
- Future work will try to include GPUs & Larrabee

```
fn      $3, $7, $13, $4  
fn      $25, $3, 1  
fn      $5, $2, $25, $2  
cgti    $20, $5, -1  
selb    $78, $3, $25, $28  
fn      $25, $21, $77  
fn      $21, $78, $77  
nop     127  
brz     $66, 180  
ori     $22, $26, 0  
fnbr    $13, 0  
ai      $15, $52, $48  
highlo  $20, $54, 0  
192  
198  
182  
fn      $6, $32, $23  
freat   $7, $16  
nop     127  
hbbr    0, 1  
fa      $37, $37, $8  
lq      $26, $13, $48  
fa      $36, $36, $21  
freat   $72, $14  
a       $78, $43, $13  
hbbr    192, 198  
ai      $12, $16, $7  
ai      $11, $42, $15  
ai      $23, $14, $77  
fn      $79, $41, $16  
fn      $5, $2, $58  
fn      $35, $36, $26  
hbbr    0, 2  
ai      $20, $20, 1  
fn      $16, $12, $19  
fn      $19, $31, $24  
fn      $8, $12, $17  
fnop    $12, $30, $27  
fn      $5, $13, $48  
fn      $22, $12, $18  
lq      $10, $13, $39  
fn      $14, $6, $23  
highlo  $23, $22, 0  
fn      $18, $27, $28  
fn      $3, $19, $19  
fn      $7, $10, $21  
fn      $26, $44, $14  
fn      $21, $17, $17, $51  
fnop    $9, $13, $39  
fn      $8, $8, $26  
lq      $28, $13, $38  
fn      $14, $18, $18, $21  
fn      $21, $16, $26  
fn      $24, $28, $25  
fn      $25, $27, $28  
nop     127  
freat   $4, $14  
fn      $77, $14, $4  
fnop    $24, $13, $38  
ai      $13, $13, 16  
lq      $22, 0($15)  
ai      $15, $15, 16  
lq      $26, $33, $79  
lq      $24, $33, $78  
lq      $28, $33, $11  
freat   $7, $77  
fn      $16, $77, $77  
192  
180  
fnbr    $28, 198  
ori     $23, $22, 0  
180  
fn      $15, $32, $23  
freat   $20, $16  
fn      $11, $30, $27  
freat   $12, $14  
fn      $22, $31, $28  
lq      $5, 0($58)  
ai      $53, $53, 1  
fn      $23, $34, $47  
fn      $13, $16, $20  
fn      $31, $34, $46  
fn      $78, $14, $12  
fn      $30, $34, $45  
fn      $7, $17, $13
```

Two Topics in One

- Accelerators (Accel)
...this is going to hurt...
- Heterogeneous systems (Hetero)
...kill me now...
- Goal of work... take away the pain and make code portable
- Code examples

```
fn      $3, $7, $13, $4
fn      $25, $3, 1
fn      $5, $2, $25, $27
cgti    $20, $5, -1
selb    $78, $3, $25, $28
fn      $25, $21, $77
fn      $21, $78, $77
nop     127
brz     $66, 198
ori     $22, $26, 0
fnbr    $13, 0
ai      $15, $52, $48
highb   $20, $54, 0
192     br      182
198     ori     $22, $26, 0
182     fn      $5, $32, $23
fn      $7, $16
nop     127
hbrz    0, 1
fa      $37, $37, $8
lq      $2, $13, $48
fa      $36, $36, $21
fn      $72, $14
a       $78, $43, $15
hbrz    192, 198
fi      $12, $16, $7
a       $11, $42, $15
fa      $23, $14, $77
a       $79, $41, $15
fa      $5, $2, $58
fs      $35, $35, $25
hbrz    0, 2
ai      $20, $20, 1
fn      $16, $12, $19
fs      $19, $31, $24
fn      $8, $12, $17
190     fn      $12, $30, $27
fs      $5, $13, $48
fn      $22, $12, $18
lq      $10, $13, $39
fn      $14, $6, $23
highb   $23, $22, 0
fs      $18, $27, $28
fn      $3, $19, $19
fs      $7, $10, $21
fn      $26, $44, $14
fn      $21, $17, $17, $51
stq     $9, $13, $39
fn      $8, $8, $26
lq      $28, $13, $38
fn      $14, $18, $18, $21
ai      $21, $16, $26
fn      $24, $28, $25
fn      $25, $27, $28
nop     127
fn      $4, $14
fi      $77, $14, $4
stq     $24, $13, $38
ai      $13, $13, 16
lq      $22, 0($15)
ai      $15, $15, 16
lq      $26, $33, $79
lq      $24, $33, $78
lq      $28, $33, $11
fn      $7, $77
fi      $16, $77, $57
192     hbrz    $28, 198
180     ori     $23, $22, 0
fn      $15, $32, $23
fn      $30, $16
fs      $11, $30, $27
fn      $12, $14
fs      $22, $31, $24
lq      $5, 0($58)
ai      $53, $53, 1
cvt     $23, $34, $47
fi      $13, $16, $20
cvt     $31, $34, $46
fi      $78, $14, $12
cvt     $30, $34, $45
fn      $7, $17, $13
```

Why Use Accelerators?

- Performance

```
fn      $3 $7 $13 $4  
fn      $25 $3 1  
fn      $5 $2 $25 $27  
cgti    $20 $5 -1  
selb    $78 $3 $25 $28  
fn      $25 $21 $77  
fn      $21 $78 $77  
nop     127  
brz     $66 198  
ori     $22 $26 0  
fnbr    $13 0  
ai      $15 $52 $48  
highb   $20 $54 0  
192:   br      182  
198:   ori     $22 $26 0  
182:   fn      $5 $32 $23  
fn      $7 $16  
nop     127  
hbrz    0 1  
fa      $37 $37 $8  
lqr     $2 $13 $48  
fa      $36 $36 $21  
fn      $72 $14  
a       $78 $43 $15  
hbrz    192 198  
fi      $12 $16 $7  
a       $11 $42 $15  
fa      $23 $14 $77  
a       $79 $41 $15  
fa      $5 $2 $58  
fs      $35 $35 $25  
hbrz    0 2  
ai      $20 $20 1  
fn      $16 $12 $19  
fs      $19 $31 $24  
fn      $8 $12 $17  
lnop    $12 $30 $27  
fs      $5 $13 $48  
fn      $22 $12 $18  
lqr     $10 $13 $39  
fn      $14 $6 $23  
highb   $23 $22 0  
fs      $18 $27 $28  
fn      $3 $19 $19  
fs      $7 $10 $21  
fn      $26 $44 $14  
fn      $21 $17 $17 $51  
stqr    $9 $13 $39  
fn      $8 $8 $26  
lqr     $28 $13 $38  
fn      $14 $18 $18 $21  
fn      $21 $16 $26  
fs      $24 $28 $25  
fn      $25 $27 $26  
nop     127  
frequent $4 $14  
fi      $77 $14 $4  
stqr    $24 $13 $38  
ai      $13 $13 16  
lqr     $22 0($15)  
ai      $15 $15 16  
lqr     $26 $33 $79  
lqr     $24 $33 $78  
lqr     $28 $33 $11  
fn      $7 $77  
fi      $16 $77 $57  
192:   hbrz    $28 198  
180:   ori     $23 $22 0  
fn      $15 $32 $23  
fn      $30 $16  
fs      $11 $38 $27  
fn      $12 $14  
fs      $22 $31 $24  
lqr     $5 0($58)  
ai      $53 $53 1  
cwr     $23 $34 $47  
fi      $13 $16 $28  
cwr     $31 $34 $46  
fi      $78 $14 $12  
cwr     $38 $34 $45  
fn      $7 $17 $13
```

Why Not Use Accelerators?

- Hard to program
 - Many architecturally specific details
 - Different ISAs between core types
 - Explicit DMA transactions to transfer data to/from the SPEs' local stores
 - Scheduling of work and communication
 - Code is not trivially portable
 - Structure of code on an accelerator often does not match that of a commodity architecture
 - Simple re-compile not sufficient

```
fn  $3, $7, $13, $9
al  $25, $3, 1
fn  $5, $2, $25, $20
cgti $20, $5, -1
selb $78, $3, $25, $28
fn  $25, $21, $77
fn  $21, $78, $77
    127
    $66, 198
    $22, $26, 0
    $13, 0
    $15, $52, $48
    $20, $54, 0
    $22, $26, 0
    $22, $26, 0
    $5, $32, $23
    $7, $16
    127
    $1, 1
    $17, $37, $8
    $2, $13, $48
    $36, $36, $21
    $72, $14
    $78, $43, $15
    $92, 198
    $12, $16, $7
    $11, $42, $15
    $23, $14, $77
    $79, $41, $16
    $5, $2, $58
    $35, $35, $25
    $2
    $20, $20, 1
    $16, $12, $19
    $19, $31, $24
    $8, $12, $17
    $12, $30, $27
    $5, $13, $48
    $2, $12, $18
    $10, $13, $39
    $14, $6, $23
    $23, $22, 0
    $18, $27, $28
    $31, $19, $19
    $71, $10, $21
    $26, $44, $14
    $21, $17, $17, $51
    $9, $13, $39
    $8, $8, $26
    $28, $13, $38
    $14, $18, $18, $21
    $21, $16, $26
    $24, $28, $25
    $25, $27, $28
    nop
    $4, $14
    $77, $14, $4
    $24, $13, $38
    $13, $13, 16
    $22, 0($15)
    $3, $16
    $33, $79
    $17, $13, $78
    $28, $33, $11
    $7, $77
    $16, $77, $37
    $28, 198
    $23, $22, 0
    $15, $32, $23
    $20, $16
    $11, $30, $27
    $12, $14
    $22, $31, $24
    $5, 0($58)
    $53, $53, 1
    $23, $34, $47
    $13, $16, $20
    $31, $34, $46
    $78, $14, $12
    $30, $34, $45
    $7, $17, $13
```

Extensions Charm++

- Added extensions
 - Accelerated entry methods
 - Accelerated blocks
 - SIMD instruction abstraction
- Extensions should be portable between architectures

```
fn      $3, $7, $13, $4
ai      $25, $3, 1
fn      $5, $2, $25, $27
cgti    $20, $5, -1
selb    $78, $3, $25, $28
fn      $25, $21, $77
fn      $21, $78, $77
nop     127
brz     $66, 198
ori     $22, $26, 0
fnbr    $13, 0
ai      $15, $52, $48
highb   $20, $54, 0
192:    br      182
198:    ori     $22, $26, 0
182:    fn      $5, $32, $23
fn      $7, $16
nop     127
hbr     0, 1
fa      $37, $37, $8
lq      $2, $13, $48
fa      $36, $36, $21
fn      $77, $14
a       $78, $43, $15
hbr     192, 198
fi      $12, $16, $7
a       $11, $42, $15
fa      $23, $14, $77
a       $79, $41, $15
fa      $5, $2, $8
fs      $35, $35, $25
hbr     0, 2
ai      $20, $20, 1
fn      $16, $12, $19
fs      $19, $31, $24
fn      $8, $12, $17
fnop    $17, $30, $27
fs      $5, $13, $48
fn      $22, $12, $18
lq      $10, $13, $39
fn      $14, $6, $23
highb   $23, $22, 0
fs      $18, $27, $28
fn      $3, $19, $19
fs      $7, $10, $21
fn      $26, $44, $14
fn      $21, $17, $17, $51
fnbr    $9, $13, $39
fn      $8, $8, $26
lq      $28, $13, $38
fn      $14, $18, $18, $21
fn      $21, $16, $26
fa      $24, $28, $25
fn      $25, $27, $26
nop     127
fn      $4, $14
fi      $77, $14, $4
stop    $24, $13, $38
ai      $13, $13, 16
lq      $22, 0($15)
ai      $15, $15, 16
lq      $26, $33, $79
lq      $24, $33, $78
lq      $28, $33, $11
fn      $7, $77
fi      $16, $77, $57
192:    hbr     $28, 198
180:    ori     $23, $22, 0
fn      $15, $32, $23
fn      $30, $16
fn      $11, $30, $27
fn      $12, $14
fn      $22, $31, $24
lq      $5, 0($58)
ai      $53, $53, 1
cvt     $23, $34, $47
fi      $13, $16, $20
cvt     $31, $34, $46
fi      $78, $14, $12
cvt     $30, $34, $45
fn      $7, $17, $13
```

Accelerated Entry Methods

- Executed on accelerator if present
- Targets computationally intensive code
- Structure based on standard entry methods
 - Data dependencies expressed via messages
 - Code is self-contained
- Managed by the runtime system
 - DMAs automatically overlapped with work on the SPEs
 - Scheduled (based on data dependencies: messages, objects)
 - Multiple independently written portions of code share the same SPE (link to multiple accelerated libraries)

Accel Entry Method Structure

```
entry [accel] void entryName  
    ( ...passed parameters... )  
    [ ...local parameters... ]  
    { ... function body ... }  
    callback_member_funcion;
```

```
objProxy.entryName( ... passed parameters ...)
```

```
fn  $3 $7 $13 $9  
ai  $25 $3 1  
fn  $5 $2 $25 $9  
cgti $20 $5 -1  
sell $78 $3 $25 $28  
fn  $25 $21 $77  
fn  $21 $78 $77  
nop  127  
hbr  $66 180  
$22 $26 0  
f  $13 0  
$15 $52 $8  
$20 $54 0  
195  
198  
182  
fn  $6 $32 $28  
freat $7 $16  
nop  127  
hbr  0 1  
fa  $37 $37 $8  
lq  $2 $13 $48  
fa  $36 $36 $21  
freat $72 $14  
a  $78 $43 $15  
hbr  192 198  
f  $12 $16 $7  
a  $11 $42 $15  
fa  $23 $14 $77  
a  $79 $41 $15  
fa  $5 $2 $58  
f  $35 $35 $25  
hbr  0 2  
ai  $20 $20 1  
fn  $16 $12 $19  
f  $19 $31 $24  
fn  $8 $12 $17  
fnop  127  
fa  $19 $30 $27  
f  $5 $13 $48  
fn  $22 $12 $18  
lq  $10 $13 $39  
fn  $14 $6 $23  
hbr  $23 $22 0  
f  $18 $27 $28  
fn  $3 $19 $19  
f  $7 $10 $21  
fn  $26 $44 $14  
fn  $21 $17 $17 $51  
freat $9 $13 $39  
fn  $8 $8 $26  
lq  $28 $13 $38  
fn  $14 $18 $18 $21  
fn  $21 $16 $26  
f  $24 $28 $25  
fn  $25 $27 $26  
nop  127  
freat $4 $14  
f  $77 $14 $4  
freat $24 $13 $38  
ai  $13 $13 16  
lq  $22 0($15)  
ai  $15 $15 16  
lq  $26 $33 $79  
lq  $24 $33 $78  
fn  $28 $33 $77  
197  
180  
hbr  $28 198  
ori  $23 $22 0  
fn  $15 $32 $28  
freat $30 $16  
f  $11 $30 $27  
freat $12 $14  
f  $22 $31 $28  
lq  $5 0($58)  
ai  $53 $53 1  
cvt  $23 $34 $47  
f  $13 $16 $20  
cvt  $31 $34 $46  
f  $78 $14 $12  
cvt  $30 $34 $45  
fn  $7 $17 $13
```


Accelerated Blocks

- Additional code that is accessible to accelerated entry methods
 - #include directives
 - Functions called by accelerated entry methods

```
fn $3,$7,$13,$9
fn $25,$3,$1
fn $5,$2,$25,$9
cgti $20,$5,-1
sellb $78,$3,$25,$28
fn $25,$21,$77
fn $21,$78,$77
nop $127
brz $66,$180
ori $22,$26,0
fnb $13,0
ai $15,$52,$48
highb $20,$54,0
192:
br $182
198:
nop $22,$26,0
182:
fn $5,$32,$23
fn $7,$16
nop $127
hbr $1,1
fa $37,$37,$58
lq $2,$13,$48
fa $36,$36,$21
fn $77,$14
a $78,$43,$15
hbr $192,$198
fi $12,$16,$7
a $11,$42,$15
fa $23,$14,$77
a $79,$41,$15
fa $5,$2,$58
fs $35,$35,$25
hbr $1,2
ai $20,$20,1
fn $16,$12,$19
fs $19,$31,$24
fn $8,$12,$17
fnop
fs $12,$30,$27
stq $5,$13,$48
fn $22,$12,$18
lq $10,$13,$39
fn $14,$6,$23
highb $23,$22,0
fs $18,$27,$28
fn $31,$19,$19
fs $7,$10,$21
fn $26,$44,$14
fn $21,$17,$17
stq $9,$13,$39
fn $8,$8,$26
lq $28,$13,$38
fn $14,$18,$18
fn $21,$16,$26
fs $24,$28,$25
fn $25,$27,$26
nop $127
fn $4,$14
fs $77,$14,$4
stq $24,$13,$38
ai $13,$13,$16
lq $22,$0($15)
ai $15,$15,$16
lq $26,$33,$79
lq $24,$33,$78
lq $28,$33,$11
fn $7,$77
fi $16,$77,$57
192:
hbr $198
ori $23,$22,0
180:
fn $15,$32,$23
fn $20,$16
fs $11,$30,$27
fn $12,$14
fs $22,$31,$24
lq $5,$0($58)
ai $53,$53,1
cui $23,$34,$47
fi $13,$16,$20
cui $31,$34,$46
fi $78,$14,$12
cui $30,$34,$45
fn $7,$17,$13
```

SIMD Abstraction

- Abstract SIMD instructions supported by multiple architectures
 - Currently adding support for: SSE (x86), AltiVec (PowerPC; PPE), SIMD instructions on SPEs
 - Generic C implementation when no direct architectural support is present
 - Types: vec4f, vec2lf, vec4i, etc.
 - Operations: vadd4f, vmul4f, vsqrt4f, etc.



“HelloWorld” Code

hello.ci

```
-----  
mainmodule hello {  
  ...  
  accelblock {  
    void sayMessage(char* msg,  
                    int thisIndex,  
                    int fromIndex) {  
      printf("%d told %d to say W\"%sW\"Wn",  
            fromIndex, thisIndex, msg);  
    }  
  };  
  
  array [1D] Hello {  
    entry Hello(void);  
    entry [accel] void saySomething(  
      int msgLen,  
      char msg[msgLen],  
      int fromIndex ) [  
      readonly : int thisIndex <impl_obj->thisIndex>  
    ] {  
      sayMessage(msg, thisIndex, fromIndex);  
    } saySomething_callback;  
  };  
};
```

Hello.C

```
-----  
class Main : public CBase_Main {  
  Main(CkArgMsg* m) {  
    CkPrintf("Running Hello on %d processors for %d elementsWn",  
            CkNumPes(), nElements);  
    char *msg = "Hello from Main";  
    arr[0].saySomething(strlen(msg) + 1, msg, -1);  
  };  
  void done(void) { CkPrintf("All doneWn"); CkExit(); };  
};  
  
class Hello : public CBase_Hello {  
  void saySomething_callback() {  
    if (thisIndex < nElements - 1) {  
      char msgBuf[128];  
      int msgLen = sprintf(msgBuf, "Hello from %d", thisIndex) + 1;  
      thisProxy[thisIndex+1].saySomething(msgLen, msgBuf,  
                                          thisIndex);  
    } else {  
      mainProxy.done();  
    }  
  };  
};
```


MD Example Code

- List of particles evenly divided into equal sized *patches*
 - Compute objects calculate forces
 - Coulomb's Law
 - Single precision floating-point
 - Patches sum forces and update particle data
 - All particles interact with all other particles each timestep
- ~92K particles (similar to ApoA1 benchmark)
- Uses SIMD abstraction for all versions

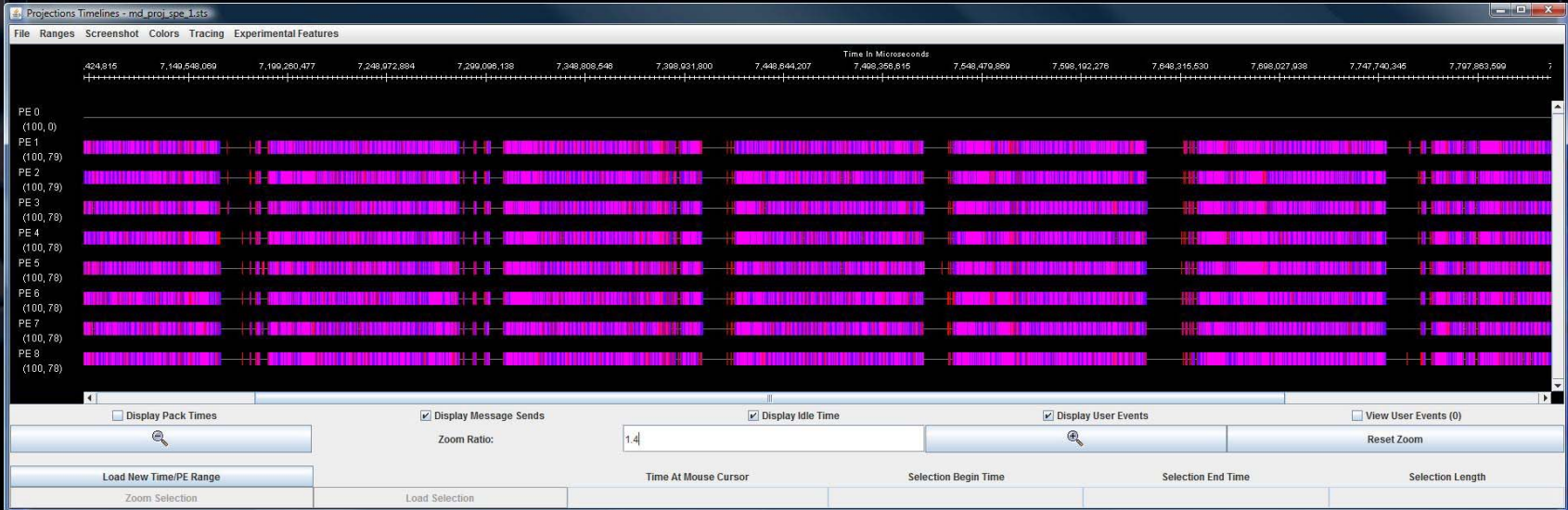
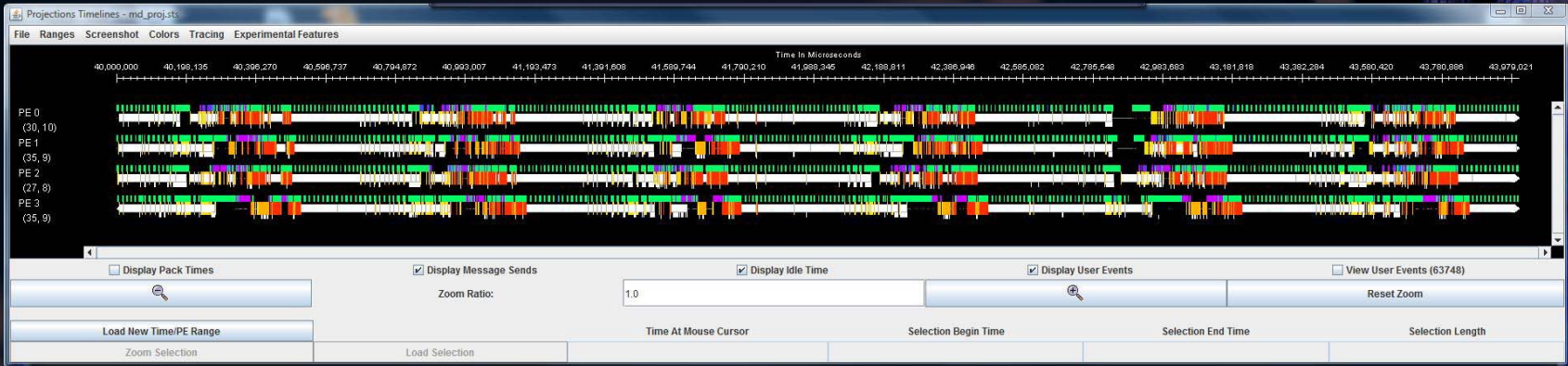
```
fr  $3.57.513.34
ai  $25.53.1
fna  $5.52.525.307
cgti  $20.55.-1
selb  $78.53.525.528
fr  $25.521.577
fr  $21.578.577
nop  127
br  $66.188
ori  $22.526.0
fmb  $13.0
ai  $15.552.48
highb  $20.554.0
192  br  182
198  ori  $22.526.0
182  fr  $6.532.528
freat  $7.516
nop  127
hbry  0 1
fr  $17.537.58
ai  $2.513.548
fna  $36.536.521
freat  $72.514
a  $78.543.515
hbry  192.198
fr  $12.516.57
ai  $11.542.515
fr  $23.514.577
ai  $79.541.515
fr  $5.52.58
fs  $35.535.525
hbry  0 2
ai  $20.528.1
fr  $16.512.519
fs  $19.531.524
fr  $8.512.517
1nop  $17.538.527
fr  $5.5131.548
1fr  $22.512.518
lqd  $10.513.539
fr  $14.56.523
highb  $23.522.0
fr  $18.527.528
fr  $3.519.519
fr  $9.510.521
fr  $26.544.514
fna  $21.517.517
stq  $9.513.539
fr  $8.58.526
lqd  $28.513.538
fna  $14.518.518
fr  $21.516.526
fr  $24.528.525
fr  $25.527.526
nop  127
freat  $4.514
fr  $77.514.54
stq  $24.513.538
ai  $13.513.16
lqd  $22.0(58)
fr  $15.515.16
lqd  $26.533.579
fr  $24.533.578
fr  $28.533.511
freat  $7.577
fr  $16.577.57
192  hbry  $28.198
180  ori  $23.522.0
fr  $15.532.528
freat  $20.516
fr  $11.538.527
freat  $12.514
fr  $22.531.528
lqd  $5.0(58)
ai  $53.553.1
cub  $23.534.547
fr  $13.516.528
cub  $31.534.546
fr  $78.514.512
cub  $38.534.545
fr  $7.517.513
```

MD Example Code

- Speedups (vs. 1 x86 core using SSE)
 - 6 x86 cores: 5.89
 - 1 QS20 chip (8 SPEs): 5.74
- GFlops/sec for 1 QS20 chip
 - 50.1 GFlops/sec observed (24.4% peak)
 - Nature of code (single inner-loop iteration)
 - Inner-loop: 124 Flops using 54 instructions in 56 cycles
 - Sequential code executing continuously can achieve, at most, 56.7 GFlops/sec (27.7% peak)
 - We observe 88.4% of the *ideal GFlops/sec for this code*
 - 178.2 GFlops/sec using 4 QS20s (net-linux layer)

```
fn 58 579 577
fn 53 57 513 58
al 525 58 1
fn 55 52 525 500
cgti 520 55 -1
selb 578 53 525 528
fn 525 521 577
fn 521 578 577
nop 127
br 566 188
ori 522 526 0
fn 513 0
al 515 552 48
high 520 554 0
192
198
182
fn 56 532 528
fn 57 516
nop 127
hbr 0 1
fn 537 537 58
lq 52 513 548
fn 536 536 521
fn 572 514
fn 578 543 515
hbr 192 198
fn 512 516 57
al 511 542 515
fn 523 514 577
al 579 541 515
fn 55 52 58
fn 535 535 525
hbr 0 2
al 520 520 1
fn 516 512 519
fn 519 531 524
fn 58 512 517
nop
fn 512 530 527
fn 55 513 548
fn 522 512 518
lq 510 513 539
fn 514 56 523
high 523 522 0
fn 518 527 528
fn 535 519 519
fn 575 510 521
fn 526 544 514
fn 521 517 517 53
fn 59 513 539
fn 58 58 526
fn 528 513 538
fn 514 518 518 521
fn 521 516 526
fn 524 528 525
fn 525 527 528
nop 127
fn 54 514
fn 577 514 54
fn 524 513 538
fn 513 513 16
fn 522 0 515
al 515 515 16
lq 526 533 579
lq 524 533 578
lq 528 533 511
fn 57 577
fn 516 577 57
fn 528 198
fn 523 522 0
fn 515 532 528
fn 520 516
fn 511 530 527
fn 512 514
fn 522 531 528
lq 55 0 558
al 553 553 1
fn 523 534 547
fn 513 516 520
fn 531 534 546
fn 528 514 512
fn 530 534 545
fn 57 517 513
```

Projections



Hetero System View

Model

Application Code
(written using the Charm++ programming model)

Software

Application Binary
(compiled for architecture 1)

Charm++ RTS
(for architecture 1)

Application Binary
(compiled for architecture 2)

Charm++ RTS
(for architecture 2)

Application Binary
(compiled for architecture N)

Charm++ RTS
(for architecture N)

Hardware

Architecture 1

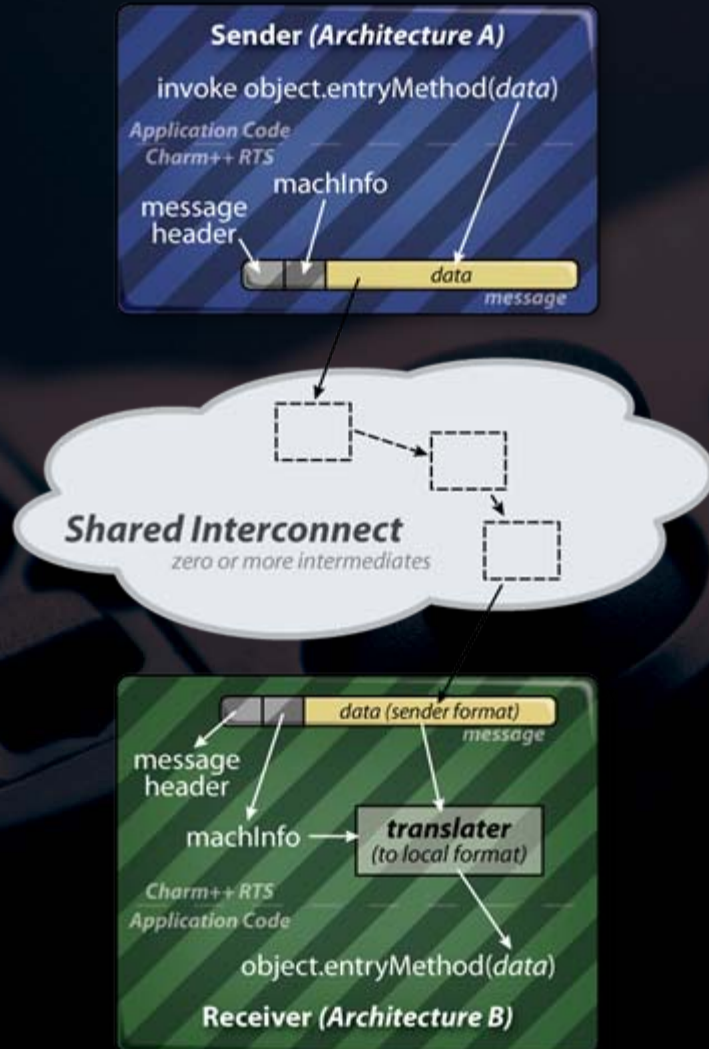
Architecture 2

Architecture N

Shared Interconnect

```
fn  $8 $79 $77
fn  $3 $7 $13 $4
fn  $25 $3 $1
fn  $5 $2 $25 $9
cgti $20 $5 -1
sellb $78 $3 $25 $28
fn  $25 $21 $77
fn  $21 $78 $77
nop  $127
br   $66 $180
ori  $22 $26 $0
fn  $13 $0
ai   $15 $52 $48
high $20 $54 $0
l92  $0
l98  $0
l82  $0
fn  $5 $32 $23
fn  $7 $16
nop  $127
hbey $0 $1
fn  $39 $37 $8
lq   $2 $13 $48
fa   $36 $36 $21
fn  $72 $14
a    $78 $43 $15
hbr  $92 $190
ai   $12 $16 $7
ai   $11 $42 $15
fa   $23 $14 $77
fn  $79 $41 $15
fn  $5 $2 $58
l    $35 $35 $26
lbbp $0 $2
fn  $26 $31 $14
fn  $21 $17 $17
l92  $0
l80  $0
fn  $15 $32 $23
fn  $20 $16
fn  $11 $38 $27
fn  $12 $14
fn  $22 $31 $24
lq   $5 $0($58)
ai   $53 $53 $1
cun  $23 $34 $47
fj   $13 $16 $28
cun  $31 $34 $46
fj   $78 $14 $12
cun  $38 $34 $45
fn  $7 $17 $13
```

Messages Across Architectures



- Makes use of Pack-UnPack (PUP) routines
 - Object migration and parameter marshaled entry method are the same as before
 - Custom pack/unpack routines for messages can use PUP framework
- Supported machine-layers:
 - net-linux
 - net-linux-cell

Making Hetero Runs

- Launch using *charmrun*
 - Compile separate binary for each architecture
 - Modified nodelist files to specify correct binary based on architecture

```
fn  $31 $7 $13 $34
fn  $25 $31
fn  $5 $2 $25 $30
cgti $20 $5 -1
sellb $78 $31 $25 $28
fn  $25 $21 $77
fn  $21 $78 $77
nop  127
br  $66 $180
ori  $22 $26 0
fn  $13 0
ai  $15 $52 $48
chighb $20 $54 0
192  br  $182
198  ori  $22 $26 0
182  fn  $5 $2 $25 $30
fn  $27 $16
nop  127
hbbr  0 1
fa  $37 $37 $8
lq  $2 $13 $48
fa  $36 $36 $21
fn  $72 $14
a  $78 $43 $15
hbr  $92 $198
fi  $12 $16 $7
fi  $11 $42 $15
fi  $35 $35 $25
hbbr  0 2
ai  $20 $20 1
ai  $17 $17 $19
ai  $24
1nop  $12 $30 $27
cgt  $5 $13 $48
fn  $22 $12 $18
lq  $10 $13 $39
fn  $14 $6 $23
chighb $23 $22 0
fi  $18 $27 $28
fn  $31 $19 $19
fi  $7 $10 $21
fn  $26 $44 $14
fn  $21 $17 $17 $51
cgt  $9 $13 $39
fn  $8 $8 $26
lq  $28 $13 $38
fn  $14 $18 $18 $21
fn  $21 $16 $26
fi  $24 $28 $25
fn  $25 $27 $26
nop  127
fn  $4 $14
fi  $77 $14 $4
cgt  $24 $13 $38
ai  $13 $13 $16
lq  $22 0($15)
ai  $15 $15 $16
lq  $26 $33 $79
lq  $24 $33 $78
fn  $7 $77
fi  $16 $77 $57
192  hbr  $28 $198
180  ori  $23 $22 0
fn  $15 $32 $23
fn  $30 $16
fi  $11 $30 $27
fn  $12 $14
fi  $22 $31 $24
lq  $5 0($58)
ai  $53 $53 1
cgt  $23 $34 $47
fi  $13 $16 $20
cgt  $31 $34 $46
fi  $78 $14 $12
cgt  $30 $34 $45
fn  $7 $17 $13
```

Hetero "Hello World" Example

Nodelist

```
-----  
group main ++shell "ssh -X"  
host kaleblade ++pathfix __arch_dir__ net-linux  
host blade_1 ++pathfix __arch_dir__ net-linux-cell  
host ps3_1 ++pathfix __arch_dir__ net-linux-cell
```

Accelblock change in hello.ci (just for demonstration)

```
-----  
accelblock {  
    void sayMessage(char* msg,  
                    int thisIndex,  
                    int fromIndex) {  
        #if CMK_CELL_SPE != 0  
            char *coreType = "SPE";  
        #elif CMK_CELL != 0  
            char *coreType = "PPE";  
        #else  
            char *coreType = "GEN";  
        #endif  
        printf("[%s] :: %d told %d to say \"%s\\n",  
               coreType, fromIndex, thisIndex, msg);  
    }  
};
```

Launch Command:

```
-----  
./charmrun ++nodelist ./nodelist_hetero +p3  
~/charm/__/arch_dir__/examples/charm++/cell/hello/hello 10
```

Output

```
-----  
Running Hello on 3 processors for 10 elements  
[GEN] :: -1 told 0 to say "Hello from Main"  
[SPE] :: 0 told 1 to say "Hello from 0"  
[SPE] :: 1 told 2 to say "Hello from 1"  
[GEN] :: 2 told 3 to say "Hello from 2"  
[SPE] :: 3 told 4 to say "Hello from 3"  
[SPE] :: 4 told 5 to say "Hello from 4"  
[GEN] :: 5 told 6 to say "Hello from 5"  
[SPE] :: 6 told 7 to say "Hello from 6"  
[SPE] :: 7 told 8 to say "Hello from 7"  
[GEN] :: 8 told 9 to say "Hello from 8"  
All done
```



Summary

- Development still in progress (both)
- Addition of accelerator extensions
 - Example codes in Charm++ distribution (the nightly build)
 - Achieve good performance
- Heterogeneous system support
 - Simple example codes running
 - Not in public Charm++ distribution yet

```
fn  $3, $7, $13, $4
ai  $25, $3, 1
fn  $5, $2, $25, $27
cgti $20, $5, -1
sellb $78, $3, $25, $28
fn  $25, $21, $77
fn  $21, $78, $77
nop  127
br  $66, 198
ori  $22, $26, 0
fn  $13, 0
ai  $15, $52, $48
highb $20, $54, 0
192  br  182
198  ori  $22, $26, 0
182  fn  $5, $32, $23
fn  $7, $16
nop  127
hb  11
fn  $37, $37, $8
lq  $2, $13, $48
fn  $36, $36, $21
fn  $72, $14
a  $78, $43, $15
hb  $92, 198
fi  $12, $16, $7
a  $11, $42, $15
fn  $23, $14, $77
a  $79, $41, $15
a  $5, $2, $8
fi  $35, $35, $25
hb  112
a  $20, $20, 1
fn  $16, $12, $19
fn  $19, $31, $24
fn  $8, $12, $17
fn  $12, $30, $27
a  $5, $13, $48
fn  $22, $12, $18
lq  $10, $13, $39
fn  $14, $6, $23
highb $23, $22, 0
fn  $18, $27, $28
fn  $31, $19, $19
fn  $71, $10, $21
fn  $26, $44, $14
fn  $21, $17, $17
a  $9, $13, $39
fn  $8, $8, $26
lq  $28, $13, $38
fn  $14, $18, $18
fn  $21, $16, $26
fn  $24, $28, $25
fn  $25, $27, $28
nop  127
fn  $4, $14
fi  $77, $14, $4
a  $24, $13, $38
ai  $13, $13, 16
lq  $22, 0($15)
ai  $15, $15, 16
lq  $26, $33, $79
lq  $24, $33, $78
lq  $28, $33, $11
fn  $7, $77
fi  $16, $77, $7
192  hb  $28, 198
180  ori  $23, $22, 0
fn  $15, $32, $23
fn  $20, $16
fn  $11, $30, $27
fn  $12, $14
fn  $22, $31, $24
lq  $5, 0($58)
ai  $53, $53, 1
c  $23, $34, $47
fi  $13, $16, $20
c  $31, $34, $46
fi  $78, $14, $12
c  $30, $34, $45
fn  $7, $17, $13
```

۰۰۰۰

```

$66 .L88
$22 $26.8
$13 0
$15 $52.48
$28 $54 0
$22 $21 0
$6 $32 $28
$27 $16
$37 $32 $38
$2 $13 $48
$36 $36 $21
$72 $14
$78 $43 $15
$92 $190
$12 $16 $7
$11 $42 $15
$23 $14 $77
$79 $41 $15
$5 $2 $58
$35 $35 $25
$2
$20 $20 .1
$16 $12 $19
$19 $31 $24
$8 $12 $17
$17 $30 $27
$5 $13 $48
$22 $12 $18
$10 $13 $39
$14 $6 $23
$23 $22 0
$18 $27 $28
$37 $19 $19
$9 $10 $21
$26 $44 $14
$21 $17 $17 $53
$9 $13 $39
$8 $8 $26
$28 $13 $38
$14 $18 $18 $21
$21 $16 $26
$24 $28 $25
$25 $27 $26
nop
$4 $14
$77 $14 $4
$24 $13 $38
$13 $13 16
$22 0($15)
$15 $15 16
$26 $33 $79
$24 $33 $78
$28 $33 $11
$7 $77
$16 $77 $57
L92:
$20 .L98
ori
L88:
$15 $32 $23
$20 $16
$11 $30 $27
$12 $14
$22 $31 $24
$5 0($50)
$53 $53 .1
$23 $34 $47
$13 $16 $20
$31 $34 $46
$78 $14 $12
$30 $34 $45
$7 $17 $13
```

Credits

- Work partially supported by NIH grant PHS 5 P41 RR05969-04: Biophysics / Molecular Dynamics
- Cell hardware supplied by IBM SUR grant awarded to University of Illinois
- Background Playstation controller image originally taken by “wloidi” on Flickr and modified by David Kunzman

