Advances in Kinetic Plasma Simulation with VPIC and Roadrunner

Kevin Bowers*, Brian Albright, Lin Yin, Bill Daughton, Vadim Roytershteyn, Ben Bergen and Tom Kwan
Los Alamos National Lab
* Guest Scientist
Overview

The Software
• VPIC: A 3d electromagnetic relativistic particle-in-cell simulation code

The Supercomputer
• Roadrunner: A petascale heterogeneous Cell / Opteron cluster

The Science
• Laser-Plasma Interaction in Inertial Confinement Fusion
• Laser Ion Acceleration
• Magnetic Reconnection

Magnetic reconnection simulation in 3d with collisions and open boundary conditions
Choir Preaching

Petaflops today

Exaflops in 10 years

Few experimental and observational capabilities will see a comparable increase

*Computational science well positioned for discoveries in biology, chemistry, climate, cosmology, energy, materials, plasmas ...*

Modern CPUs Optimized for Games

Floating point intensive games use small matrix / short vector ops in single precision

Single precision 4-vector SIMD (Single-Instruction-Multiple-Data) extensions common

Not optimized for traditional double precision large vector operations

$\mathbf{x}' = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

$v \cdot n = |v| |n| \cos \theta_{vn}$
Modern CPUs Optimized for Games

Floating point intensive games use small matrix / short vector ops in single precision 4-vector SIMD (Single Instruction Multiple Data) extensions common. Not optimized for traditional large vector operations.

Roadrunner based on a chip originally designed for the Sony Playstation 3 videogame console.
The Speed of Light is Too Slow

Consider a registered ECC DDR2-DIMM in a node with 3.2 GHz dual-issue 4-vector SIMD cores (e.g., Roadrunner)

Characteristic time for a signal at the effective speed of light to travel around the DIMM is ~3.2 ns

This alone is ~10 clocks
Time enough for ~80 flops / core

This is optimistic; many other delays
The Speed of Light is Too Slow

Consider a registered ECC DDR2-DIMM in a node with 3.2 GHz dual-issue 4-vector SIMD cores (e.g., Roadrunner). Characteristic time for a signal at the effective speed of light to travel around the DIMM is ~3.2 ns. This alone is ~10 clocks, time enough for ~80 fops/core. This is optimistic; many other delays are due to physical limitations, moving data between, and even within, modern microchips is more time consuming than performing computations.
Overview

The Software

• VPIC: A 3d electromagnetic relativistic particle-in-cell simulation code

Modeling capabilities

Comparison with other techniques

Implementation considerations

Helicity dissipation in astrophysical plasma
(Bowers and Li, Phys Rev Lett, 2006)
What does VPIC do?

VPIC integrates the relativistic Maxwell-Boltzmann system in a linear background medium for multiple particle species,

\[ \frac{\partial_t f_s}{f_s} + c\gamma^{-1}u \cdot \nabla f_s + \frac{q_s}{m_s c} \left( \frac{E + c\gamma^{-1}u \times B}{\vec{v}} \right) \cdot \nabla u f_s = \left( \frac{\partial_t f_s}{f_s} \right)_{\text{coll}} \]

\[ \frac{\partial_t E}{E} = \epsilon^{-1} \nabla \times \mu^{-1} B - \epsilon^{-1} J - \epsilon^{-1} \sigma E \]

\[ \frac{\partial_t B}{B} = -\nabla \times E, \]

in time with an explicit-implicit mixture of velocity Verlet, leapfrog, Boris rotation and exponential differencing based on a reversible phase-space-volume conserving 2nd order Trotter factorization.

Direct discretization of \( f_s \) is prohibitive; \( f_s \) is sampled by particles,

\[ d_t r_{s,n} = c\gamma^{-1} u_{s,n} \quad d_t u_{s,n} = \frac{q_s}{m_s c} \left( \frac{E}{r_{s,n}} + c\gamma^{-1} u_{s,n} \times B \right)_{r_{s,n}} \]

Particles obey the same Boltzmann equation outside of collisions.

A smooth \( J \) is extrapolated from the particles; as a result, \( E, B \) and \( J \) can be sampled on a mesh and interpolated to and from particles.
What does VPIC do?

VPIC integrates the relativistic Maxwell-Boltzmann system in a linear background medium for multiple particle species, in time with an explicit-implicit mixture of velocity Verlet, leapfrog, Boris rotation and exponential differencing based on a reversible phase-space-volume conserving 2nd order Trotter factorization.

Direct discretization of \( f_s \) is prohibitive; \( f_s \) is sampled by particles, and as a result, E, B and J can be sampled on a mesh and interpolated to and from particles.

Particles obey the same Boltzmann equation outside of collisions.

A smooth J is extrapolated from the particles; as a result, E, B and J can be sampled on a mesh and interpolated to and from particles.

Theoretical explanation mostly useful for making babies cry
What does VPIC really do?

Initial State

Read: 32 bytes
Write: 0 bytes
Compute: 0 flop

PIC: Particle In Cell (a.k.a. Voxel)
What does VPIC really do?

Initial State
Interpolate $E$ and $B$

Read: 72 bytes
Write: 0 bytes
Compute: 27 flop
What does VPIC really do?

Initial State
Interpolate $E$ and $B$
Update $u$

Read: 0 bytes
Write: 0 bytes
Compute: 107 flop
What does VPIC really do?

- Initial State
- Interpolate $E$ and $B$
- Update $u$
- Compute Motion

Read: 0+48 bytes
Write: 0+48 bytes
Compute: 42+70 flop
What does VPIC really do?

Initial State
Interpolate $E$ and $B$
Update $u$
Compute Motion
Update $r$ and $J$

Read: 56 bytes
Write: 48 bytes
Compute: 168 flop

Los Alamos
ASC
NASA
What does VPIC really do?

Initial State
Interpolate $E$ and $B$
Update $u$
Compute Motion
Update $r$ and $J$
Update $r$ and $J$

Read: 56 bytes
Write: 48 bytes
Compute: 168 flop
What does VPIC really do?

- Initial State
- Interpolate $E$ and $B$
- Update $u$
- Compute Motion
- Update $r$ and $J$
- Update $r$ and $J$
- Update $r$ and $J$

Read: 56 bytes
Write: 48 bytes
Compute: 168 flop
What does VPIC really do?

Initial State
Interpolate $E$ and $B$
Update $u$
Compute Motion
Update $r$ and $J$
Update $r$ and $J$
Update $r$ and $J$

Final State

Read: 0 bytes
Write: 32 bytes
Compute: 0 flop

Net Read: 152+ 56 $n_c$ bytes
Net Write: 80+ 48 $n_c$ bytes
Net Compute: 246+168 $n_c$ flop
Why use PIC?

Vlasov codes model similar equations
- But do not scale to high dimensional systems
Why use PIC?

Vlasov codes model similar equations
  • But do not scale to high dimensional systems

Traditional Monte-Carlo easy to parallelize + accelerate
  • But not suitable for time dependent effects
Why use PIC?

Vlasov codes model similar equations
• But do not scale to high dimensional systems

Traditional Monte-Carlo easy to parallelize + accelerate
• But not suitable for time dependent effects

Computational fluid dynamics cheaper
• But impossible if the equation of state is unknown
Why use PIC?

Vlasov codes model similar equations
• But do not scale to high dimensional systems

Traditional Monte-Carlo easy to parallelize + accelerate
• But not suitable for time dependent effects

Computational fluid dynamics cheaper
• But impossible if the equation of state is unknown

Molecular dynamics closely related
• But orders of magnitude more expensive ...
MD versus PIC

MD focus is short range

- Necessary when nearby interaction potential energy >> thermal energy
- Difficult for particles to represent many atoms
- \textit{Flops / particle / step large} (10^3 - 10^4)
## MD versus PIC

<table>
<thead>
<tr>
<th>MD focus is short range</th>
<th>PIC focus is long range</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Necessary when nearby interaction potential energy $&gt;&gt;$ thermal energy</td>
<td>• Useful when nearby interaction potential energy $&lt;&lt;$ thermal energy</td>
</tr>
<tr>
<td>• Difficult for particles to represent many atoms</td>
<td>• Approximates short range interactions</td>
</tr>
<tr>
<td>• <em>Flops / particle / step large</em> $(10^3 - 10^4)$</td>
<td>• <em>Flops / particle / step small</em> $(\sim 10^2)$</td>
</tr>
</tbody>
</table>
Typical VPIC Simulations

Many particles / node ($10^7 - 10^8$)
- Particle data does not fit in cache
- >90% expense is particle pushing
Typical VPIC Simulations

Many particles / node ($10^7 - 10^8$)
- Particle data does not fit in cache
- >90% expense is particle pushing

Many voxels / node ($10^4 - 10^5$)
- Field data does not fit in cache
- Many particles / voxel ($10^2 - 10^4$)
Typical VPIC Simulations

Many particles / node ($10^7$ - $10^8$)
- Particle data does not fit in cache
- >90% expense is particle pushing

Many voxels / node ($10^4$ - $10^5$)
- Field data does not fit in cache
- Many particles / voxel ($10^2$ - $10^4$)

Few voxel boundaries crossed / particle / step
- Speed of light well resolved and $v<c$
Typical VPIC Simulations

Many particles / node ($10^7 - 10^8$)
• Particle data does not fit in cache
• >90% expense is particle pushing

Many voxels / node ($10^4 - 10^5$)
• Field data does not fit in cache
• Many particles / voxel ($10^2 - 10^4$)

Few voxel boundaries crossed / particle / step
• Speed of light well resolved and $v < c$

Internode communications naturally optimal
• Communication every step, but, because of finite $c$, data needed on a node already there or nearby
Typical VPIC Simulations

Many particles / node ($10^7 - 10^8$)
- Particle data does not fit in cache
- $>90\%$ expense is particle pushing

Many voxels / node ($10^4 - 10^5$)
- Field data does not fit in cache
- Many particles / voxel ($10^2 - 10^4$)
- Few voxel boundaries crossed / particle / step
  - Speed of light well resolved and $v < c$

Internode communications naturally optimal
- Communication every step, but, because of finite $c$,
data needed on a node already there or nearby

VPIC isn’t like a matrix calculation with $O(N^3)$ compute on $O(N^2)$ data

Low compute to data motion ratio (~1 flop / byte) makes high performance hard to achieve

Performance limited by local data motion
### Bad Ideas

<table>
<thead>
<tr>
<th>Absolute particle coordinates</th>
<th><em>Destroys precision</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bits wasted resolving voxel indices</td>
</tr>
</tbody>
</table>

*Slow interpolation*

Float - int casts (or worse)
## Bad Ideas

<table>
<thead>
<tr>
<th>Absolute particle coordinates</th>
<th>Destroys precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bits wasted resolving voxel indices</td>
</tr>
<tr>
<td></td>
<td>Slow interpolation</td>
</tr>
<tr>
<td></td>
<td>Float - int casts (or worse)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unsorted particles</th>
<th>Cache misses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Field data accessed randomly</td>
</tr>
</tbody>
</table>
## Bad Ideas

<table>
<thead>
<tr>
<th>Absolute particle coordinates</th>
<th>Destroys precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsorted particles</td>
<td>Bits wasted resolving voxel indices</td>
</tr>
<tr>
<td>Slow interpolation</td>
<td>Float - int casts (or worse)</td>
</tr>
<tr>
<td>Advance done with several passes</td>
<td>Cache misses</td>
</tr>
<tr>
<td></td>
<td>Field data accessed randomly</td>
</tr>
<tr>
<td></td>
<td>Bandwidth wasted</td>
</tr>
<tr>
<td></td>
<td>Data touched several times / step</td>
</tr>
<tr>
<td>Bad Ideas</td>
<td>Details</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Absolute particle</td>
<td>Destroys precision</td>
</tr>
<tr>
<td>coordinates</td>
<td>Bits wasted resolving voxel indices</td>
</tr>
<tr>
<td></td>
<td>Slow interpolation</td>
</tr>
<tr>
<td></td>
<td>Float - int casts (or worse)</td>
</tr>
<tr>
<td>Unsorted particles</td>
<td>Cache misses</td>
</tr>
<tr>
<td></td>
<td>Field data accessed randomly</td>
</tr>
<tr>
<td>Advance done with</td>
<td>Bandwidth wasted</td>
</tr>
<tr>
<td>several passes</td>
<td>Data touched several times / step</td>
</tr>
<tr>
<td>Each component</td>
<td>Bandwidth wasted</td>
</tr>
<tr>
<td>stored in own array</td>
<td>Small unaligned accesses</td>
</tr>
</tbody>
</table>
## Bad Ideas

| Absolute particle coordinates | Destroys precision | Bits wasted resolving voxel indices |
| Unsorted particles            | Slow interpolation | Float - int casts (or worse)      |
| Advance done with several passes | Bandwidth wasted | Data touched several times / step |
| Each component stored in own array | Bandwidth wasted | Small unaligned accesses |
| Field samples used for interpolation | Too few “ways” to keep track | 29 diff memory regions accessed / particle |
Bad Ideas

Absolute particle coordinates
Unsorted particles
Advance done with several passes
Each component stored in own array
Field samples used for interpolation
Destroys precision
Bits wasted resolving voxel indices
Slow interpolation
Float - int casts (or worse)
Cache misses
Field data accessed randomly
Bandwidth wasted
Small unaligned accesses
Too few "ways" to keep track
25 diff memory regions accessed / particle

If VPIC were implemented conventionally, ~31 physical DRAM transfers / particle / step and not many flops to show for them

Need data flow optimization techniques
### Good Ideas

<table>
<thead>
<tr>
<th>Idea</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voxel index + offset particle coordinates</td>
<td>Maximizes precision&lt;br&gt;Bits conserved; critical in single precision</td>
</tr>
<tr>
<td>Sorted particles</td>
<td>Fast interpolation&lt;br&gt;No casts; almost trivial computation</td>
</tr>
<tr>
<td>Advance done in a single pass</td>
<td>Cache hits&lt;br&gt;Field data approximately streamed</td>
</tr>
<tr>
<td>Similar components grouped together</td>
<td>Bandwidth conserved&lt;br&gt;Particle data touched once / step</td>
</tr>
<tr>
<td>Precompute voxel interpolation_coeffs</td>
<td>Bandwidth conserved&lt;br&gt;Large aligned accesses</td>
</tr>
<tr>
<td></td>
<td>Many “ways” to keep track&lt;br&gt;2 diff memory regions accessed / particle</td>
</tr>
</tbody>
</table>
No Apologies

VPIC designed with single precision in mind
• Half bytes moved and wider SIMD available
No Apologies

VPIC designed with single precision in mind
• Half bytes moved and wider SIMD available

Usually, discretization error $\gg$ single precision error
• Single precision okay if very carefully implemented
• Doubles and “numerical hygiene” used as necessary
• Extensive convergence studies and validation against theory, experiment, double precision codes
No Apologies

VPIC designed with single precision in mind
• Half bytes moved and wider SIMD available

Usually, discretization error $\gg$ single precision error
• Single precision okay if very carefully implemented
• Doubles and “numerical hygiene” used as necessary
• Extensive convergence studies and validation against theory, experiment, double precision codes

Stabilized to the point where each voxel has identical numerical properties regardless how the voxel mesh is translated, oriented or reflected
No Apologies

VPIC designed with single precision in mind
• Half bytes moved and wider SIMD available
Usually, discretization error >> single precision error
• Single precision okay if very carefully implemented
• Doubles and "numerical hygiene" used sporadically
• Extension convergence studies and validation
  against theory, experiment, double precision codes
Stabilized to the point where each voxel has identical
numerical properties regardless how the voxel mesh
is translated, oriented or reflected

**When in single precision, developers care more about arithmetic error**

Unlike double precision, ignoring it often leads to
catastrophes

We die a little bit on the inside when CPUs and compilers take
short cuts (they often do)
Overview

The Supercomputer

• Roadrunner: A petascale heterogeneous Cell / Opteron cluster

Hardware Description

Porting Details

Measured performance

Preliminary 3d Collisional VPIC Simulation of MRX (Magnetic Reconnection eXperiment)
Cell Broadband Engine

1 general purpose core, “PPE”

8 special 4-vector SIMD cores, “SPE”

Each SPE can only directly access its 256KB “local store”

Local store like cache but memory transfers explicitly managed by “MFC”
Triblade Compute Nodes

- **Opteron cores**
- **one-to-one with**
- **Cell eDPs**
- **(2 GB/s bandwidth)**

**IBM LS21 Blade**

- 8 GB DDR2-667
- Opteron 1.8 GHz
- Opteron 1.8 GHz

**IBM QS22 Blade**

- 4 GB DDR2-800
- Cell eDP 3.2 GHz

**IBM LS21 Blade**

- 8 GB DDR2-667
- Opteron 1.8 GHz
- Opteron 1.8 GHz

**IBM QS22 Blade**

- 4 GB DDR2-800
- Cell eDP 3.2 GHz

**HT2100**

**PCI-e x8**

**Infiniband 4xDDR**
Roadrunner

12,960 Opteron cores - 0.1 Pflop/s (s.p.)
12,960 Cell eDP chips - 3.0 Pflop/s (s.p.)
Porting

Observations

- Most compute in the SPEs
- SPE / Cell DRAM bandwidth (25 GB/s) >> SPE / Opteron DRAM bandwidth (2 GB/s)
- Bandwidth off-node same for Cell and Opteron (IB)
Porting

Observations

• Most compute in the SPEs
• SPE / Cell DRAM bandwidth (25 GB/s) >> SPE / Opteron DRAM bandwidth (2 GB/s)
• Bandwidth off-node same for Cell and Opteron (IB)

*Strategy*: Flatten Roadrunner

• All calculations done on Cells
• All data stored in Cell DRAM
• Opterons relay Cell communication and I/O
SPE Accelerated Particle Advance

Each SPE assigned a segment containing a multiple of 16 particles and an exclusive current accumulator.

The PPE assigned leftover particles.
SPE Accelerated Particle Advance

Each SPE assigned a segment containing a multiple of 16 particles and an exclusive current accumulator.

The PPE assigned leftover particles.

SPEs stream through segments with triple buffering in blocks of 512 particles.
SPE Accelerated Particle Advance

The heart of it all: A 512-line part read-only / part write-back software cache handles random access

- **Fully-associative:** A line can hold any voxel’s data
- **Least-recently-used:** New data evicts oldest data

The last 512 unique requests guaranteed in cache
SPE Accelerated Particle Advance

The heart of it all: A 512-line part read-only / part write-back software cache handles random access

- Fully-associative: A line can hold any voxel’s data
- Least-recently-used: New data evicts oldest data

The last 512 unique requests guaranteed in cache

cache_fetch called on all 512 particles in a new block
- Most are hits; DMA transfers started for misses
- Returns which lines will hold the voxels’ data

cache_wait then completes any pending fetches

cache_fetch non-trivial internally but a fast $O(1)$
SPE Accelerated Particle Advance

Particles processed 16 at a time
- Original x86 4-vector SIMD kernel hand unrolled and modulo scheduled by 4; register file size (128), pipeline hazards and local store limit further unrolling
SPE Accelerated Particle Advance

All these techniques result in:

**A SPE kernel that operates exclusively out of local store**

Most SPE registers used
Most local store used
All 32 DMA channels / SPE used
Most DMA transfers overlapped
Many independent SIMD instructions
Minimal DMA transfers / particle
Kernel Performance

162.0 million cold particles advanced / s / Cell
÷ 10.3 million cold particles advanced / s / Opteron

15.7x speedup
Kernel Performance

162.0 million cold particles advanced / s / Cell
÷ 10.3 million cold particles advanced / s / Opteron

15.7x speedup
÷ 1.8x faster SPE clock rate
÷ 8.0x more SPE cores than Opteron cores

1.1x clock-for-clock speedup, in spite of SPE minimalism and VPIC’s tuning for x86
Kernel Performance

162.0 million cold particles advanced / s / Cell
\div 10.3 million cold particles advanced / s / Opteron

15.7x speedup
\div 1.8x faster SPE clock rate
\div 8.0x more SPE cores than Opteron cores

1.1x clock-for-clock speedup, in spite of SPE minimalism and VPIC’s tuning for x86

0.517 P\text{flop/s on all 18 Roadrunner Connected Units}

Need 203,000 Opteron cores for similar performance
Amdahl’s Whack-a-Mole

Particle advance accelerated 15.7x

*Amdahl’s Law:*
Rest of code relatively more costly

Before

<table>
<thead>
<tr>
<th>96%</th>
<th>4%</th>
</tr>
</thead>
</table>

After

| 6%  | 4%  |

Overall Speed Up: 10x
Amdahl’s Whack-a-Mole

Particle advance accelerated 15.7x

*Amdahl’s Street Justice:*
Rest of code absolutely more costly
PPE cores less powerful than Opteron cores

Overall Speed Up:
5.6x
Amdahl’s Whack-a-Mole

Particle advance accelerated 15.7x

Amdahl’s Street Justice:
Rest of code absolutely more costly
PPE cores less powerful than Opteron cores

End-to-end performance more sensitive to unaccelerated kernels than conventional platforms. Particle sort and many field update kernels were also SPE accelerated (several fold speedups).

Amdahl bottlenecks are now frequently one-off user-provided application-specific in-situ diagnostics. User experience, improved development models needed.
End-to-End Performance

Two simulations in LPI parameter study (Albright et al, Phys Plasmas, 2008) used to benchmark weak scaling

Same physics but 10x faster

Trillion-particle simulations at 0.374 Pflop/s sustained on 17 CUs (Bowers et al, SC08)
Overview

The Science

- Laser-Plasma Interaction in Inertial Confinement Fusion
- Laser Ion Acceleration
- Magnetic Reconnection

For each, a brief overview of current research with VPIC on Roadrunner

Conclusions

Magnetic Island Detachment (Yin et al, Phys Rev Lett, 2008)
Inertial Confinement Fusion

Lasers implode a fusion fuel capsule to “ignite” it; thermonuclear burning plasma

“Minimizing laser-plasma instabilities in the NIF hohlraum is a key to achieving ignition.”
- LLNL web site
Inertial Confinement Fusion

*LPI (Laser Plasma Interaction) an issue*

- **Laser scattering:** Too little compression
- **Laser scattering:** Asymmetric compression
- **e⁻ Preheating:** Harder to compress hot plasma

LPI Nonlinear Saturation (Yin et al, Phys Rev Lett, 2007)
The Petascale Challenge

In 2010, ICF ignition experiments start at Livermore’s National Ignition Facility (NIF)

The multi-billion dollar question:
What is the risk from LPI?

Petascale computing can address this issue
Computational Science in Action

Linear theory for SRS (Stimulated Raman Scattering) in LPI developed
Drake et al, Phys Fluids, 1974

Trident experiments observe unexplained behavior
Montgomery et al, Phys Plasmas, 2002

Trident Experiments (527 nm, f/4.5 Gaussian beam, $T_e=700$ eV)

![Graph showing SRS Reflectivity vs Intensity (W/cm²)]

- High intensity saturation
- Linear estimates
- Low intensity sharp onset

$k\lambda_D=0.33$
$k\lambda_D=0.35$
Computational Science in Action

VPIC identifies key physics
Plasma wave bowing, self-focusing, filamentation and trapped particle modulational instability cause rapid onset and saturation (Yin et al, Phys Rev Lett, 2007)
Reflectivity agrees with experiment

Simulation insights lead to non-linear SRS theories
Rose and Yin, Phys Plasmas, 2008, Yin et al, Phys Plasmas, 2009

VPIC now being used on Roadrunner to understand and predict LPI in NIF

LPI in NIF f/8 laser speckle
0.4T particles, 2B voxels, 115K RR cores
Laser Ion Acceleration

High energy C$^{+6}$ beams observed from an ultra-intense short laser pulse incident on a thin foil


VPIC corroborates and discovers a process for higher energies

Relativistic effects make foil transparent for ultra-high contrast pulses and thinner foils, allowing pulse to “breakout” and accelerate ions (Yin et al, Laser and Particle Beams, 2006)
Laser Ion Acceleration

Simulation insights lead to new acceleration theories
Relativistic Buneman instability for linear polarization (Albright et al., Phys Plasmas 2007)

**VPIC prediction experimentally confirmed**
Prediction drove Trident’s redesign

Circular polarized pulse radiation pressure yields ~GeV C^{+6} ions
Conclusions

*Petascale supercomputers can change the way we do science*

Tapping the potential requires rethinking codes and analysis

Data motion is not free

*Supercomputers getting faster but not the speed of light*

Data flow optimization future proofs codes

VPIC data flow optimized almost 8 years ago yet needed no structural modifications to realize order-of-magnitude speedups on Roadrunner

*Roadrunner is a glimpse of the future*

Routine petascale computations, 100,000+ core parallelism, heterogeneous cores and intermingled compute / memory

Data flow optimization paramount
Acknowledgments

Research supported in part by the Los Alamos LDRD Program, DOE, NSF and NASA

Special thanks to IBM Roadrunner team (Cornell Wright, Bill Brandmeyer and Chris Engel) for the opportunity to use Roadrunner during early testing

Thanks to Drs. Ken Koch, Hui Li, Jeremy Margulies, Eric Nelson and Tiankai Tu for assistance with slides. Most 3d visualizations performed with EnSight Gold by CEI Inc

Work performed under the auspices of the United States Department of Energy by the Los Alamos National Security LLC Los Alamos National Laboratory under contract DE-AC52-06NA25396

Harris sheet tearing (Yin et al, Phys Rev Lett, 2008)
Physically, Cells cannot communicate directly.
Relay Library

Infiniband Network

- Relay hides traversal for transparent Cell-to-Cell communication
- Reduces hybrid complexity

OpenMPI Interface

DaCS PCIe Interface

Opteron Core

Cell eDP 1 PPE + 8 SPEs

Opteron Core

Cell eDP 1 PPE + 8 SPEs