APR16-2016-001004
Abstract for an Invited Paper
for the APR16 Meeting of
the American Physical Society

PIC codes for plasma accelerators on emerging computer architectures (GPUS, Multicore/Manycore CPUS)

HENRI VINCENTI, LBNL

The advent of exascale computers will enable 3D simulations of a new laser-plasma interaction regimes that were previously out of reach of current Petasale computers. However, the paradigm used to write current PIC codes will have to change in order to fully exploit the potentialities of these new computing architectures. Indeed, achieving Exascale computing facilities in the next decade will be a great challenge in terms of energy consumption and will imply hardware developments directly impacting our way of implementing PIC codes.\pard\pardAs data movement (from die to network) is by far the most energy consuming part of an algorithm future computers will tend to increase memory locality at the hardware level and reduce energy consumption related to data movement by using more and more cores on each compute nodes ("fat nodes") that will have a reduced clock speed to allow for efficient cooling. To compensate for frequency decrease, CPU machine vendors are making use of long SIMD instruction registers that are able to process multiple data with one arithmetic operator in one clock cycle. SIMD register length is expected to double every four years. GPU's also have a reduced clock speed per core and can process Multiple Instructions on Multiple Datas (MIMD).\pard\pardAt the software level Particle-In-Cell (PIC) codes will thus have to achieve both good memory locality and vectorization (for Multicore/Manycore CPU) to fully take advantage of these upcoming architectures. In this talk, we present the portable solutions we implemented in our high performance skeleton PIC code PICSAR to both achieve good memory locality and cache reuse as well as good vectorization on SIMD architectures. We also present the portable solutions used to parallelize the Pseudo-sepctral quasi-cylindrical code FBPIC on GPUs using the Numba python compiler.