CMB Map-Making
In The Era Of Planck

Julian Borrill
Computational Cosmology Center, Berkeley Lab
& Space Sciences Laboratory, UC Berkeley

Idealized Map-Making

$$d_t = n_t + P_{tp}\, s_p$$

- Noise – Gaussian
- Signal – Sky-synchronous

$$d_p = (P_{tp}^T\, N_{tt'}^{-1}\, P_{t'p'})^{-1}\, P_{tp}^T\, N_{tt'}^{-1}\, d_{t'}$$

$$N_{pp'} = (P_{tp}^T\, N_{tt'}^{-1}\, P_{t'p'})^{-1}$$

- How we solve this in practice depends on
  a) the actual properties of the observation data
  b) computational tractability

Noise Properties

1. (Piecewise) Stationary noise
   - Required to estimate noise statistics

2a. (Block) Circulant approximation
   - $N_{tt'}^{-1}$ is diagonal in Fourier domain
   - Solve GLS equation

2b. Destriping approximation
   - n_t is white + offset over some interval (baseline)
   - Solve for baseline offsets, subtract & bin

2c. Hybrid
   - Solve for baseline offsets & subtract
   - Solve GLS equation for resulting timestream

Signal Properties

- Violations of sky-synchronicity (s_p constant)
  - Sub-pixel structure
    - pixelize well below beam scale
  - Bandpass mismatch
  - Beam asymmetry, mismatch, bandpass variation
    - incorporate into pointing weights
      - spatially varying multi-component sky
      - beam decomposition
    - ignore & deal with map residuals

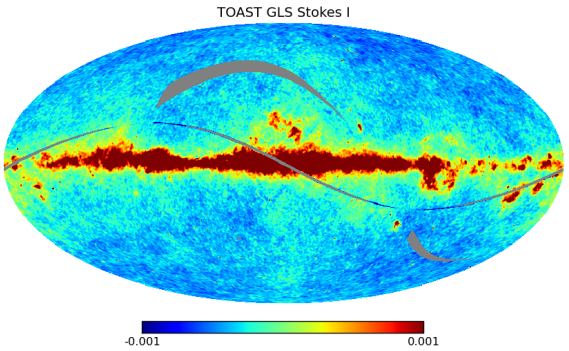
Planck Map-Making Methods

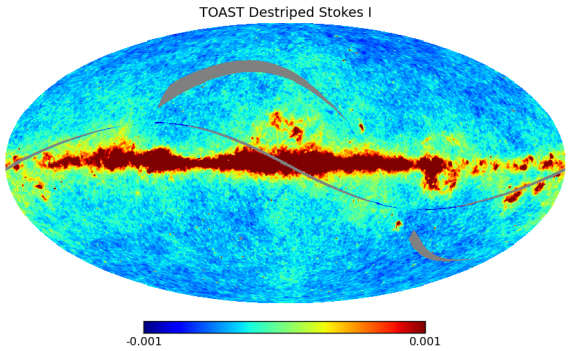
- Maximum likelihood mapmakers
  - Require accurate noise statistics
  - Require optimization to run well
- Destriping mapmakers
  - No noise model needed (unless baselines are short)
  - Typically faster (unless baselines are short)
- Extensive comparison study between (most) map-making algorithms & implementations (eg. Ashdown et al)
- First release Planck maps (March 2013) will be destriped
  - Polkapix (HFI DPC): pointing period baselines
  - MADAM (LFI DPC): 1s baselines
  - MADAM/TOAST (HFI+LFI sims – USPDC/NERSC)

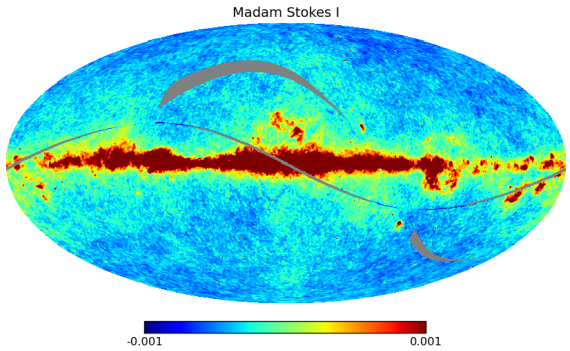
Planck Maps

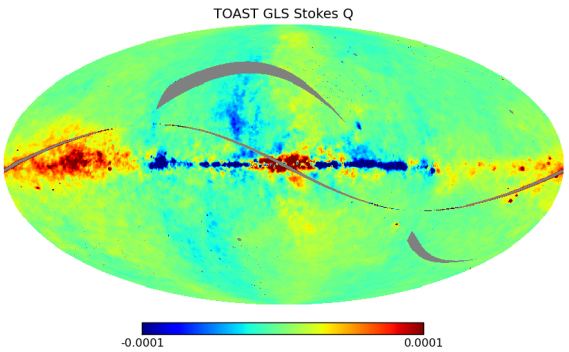
The following maps are made from **SIMULATED DATA ONLY** (but the FFP6 simulations are very realistic)

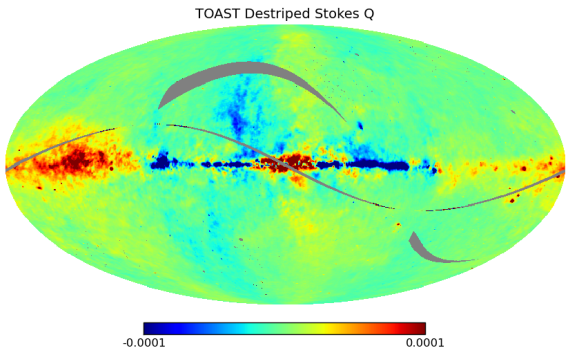
FFP6 Single Survey 30GHz IQU

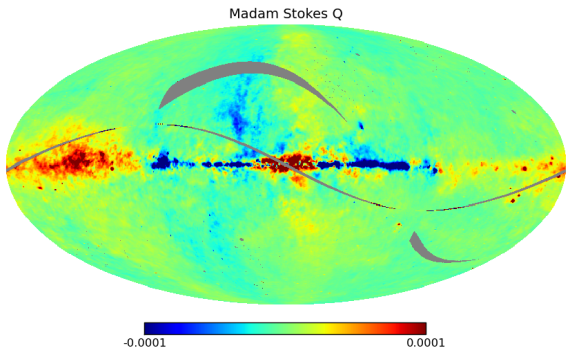


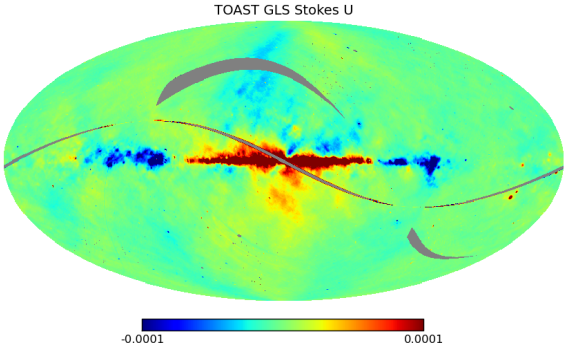


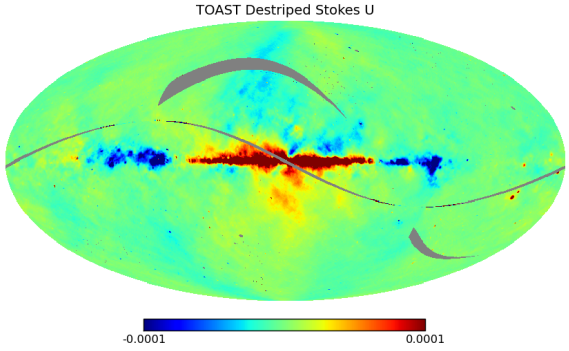


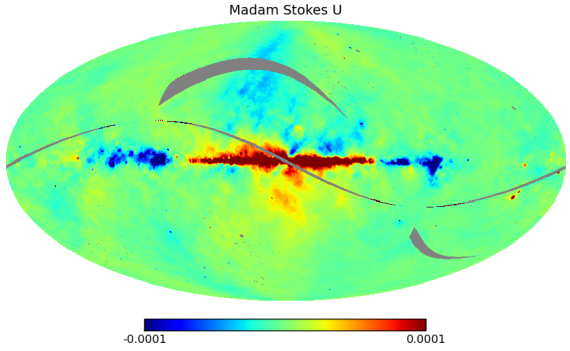




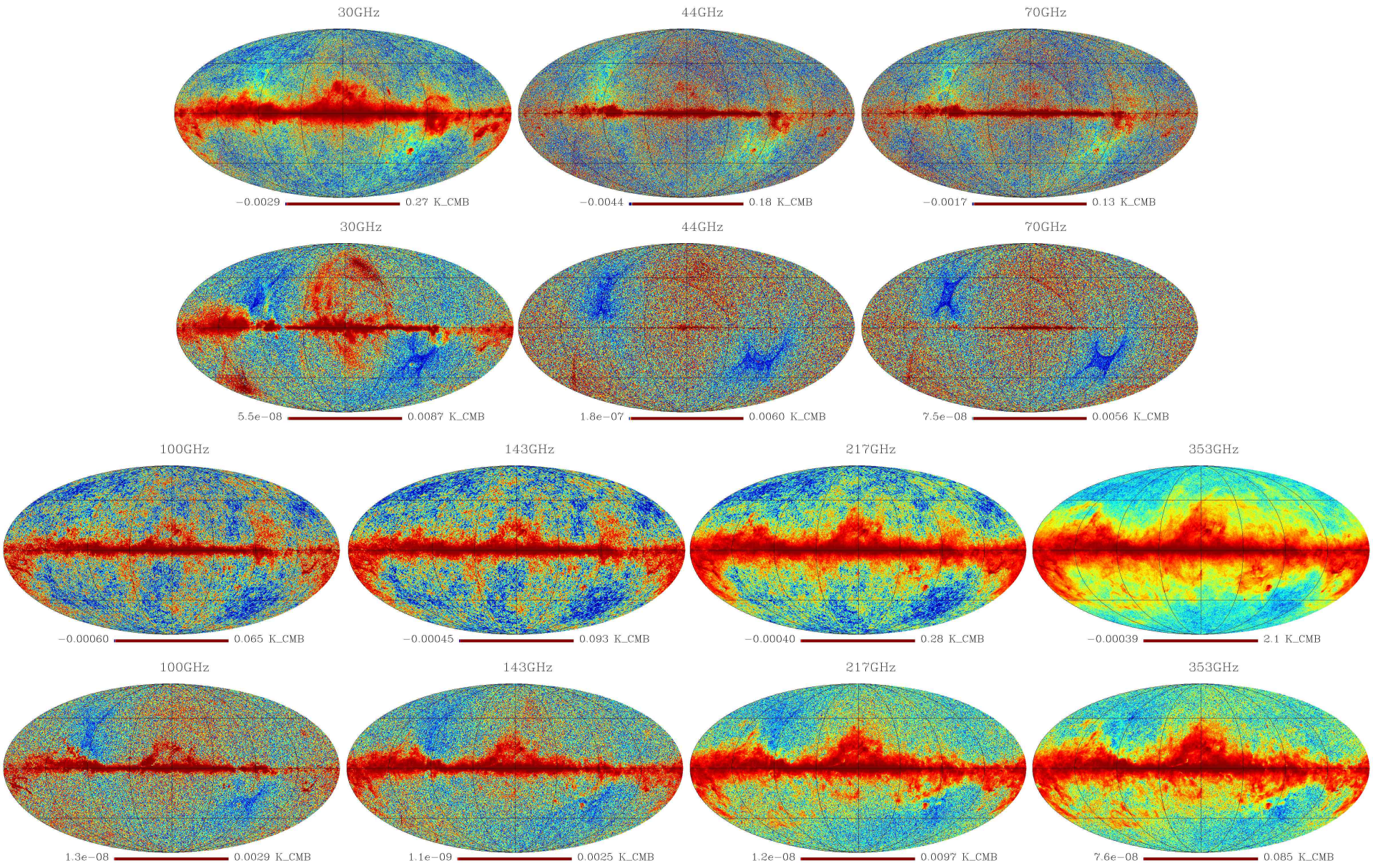




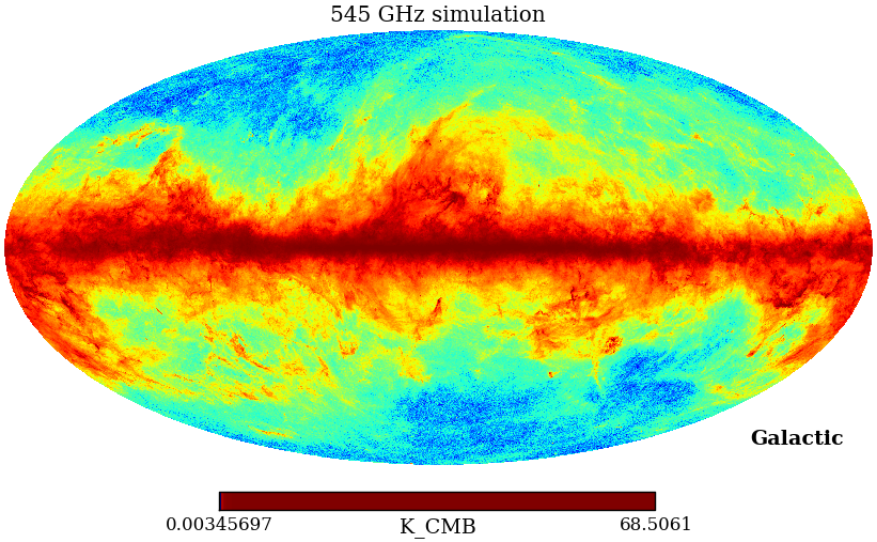


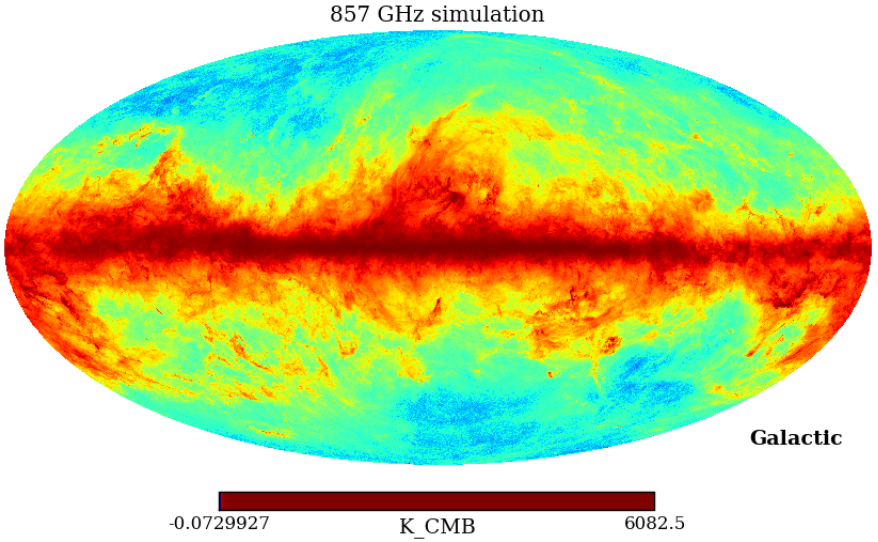


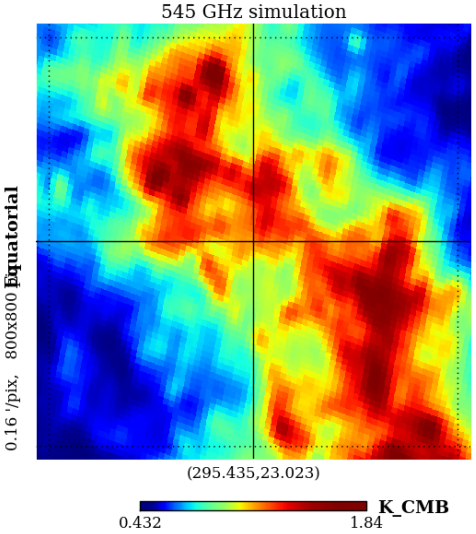
FFP6 Nominal 30-353GHz T&P

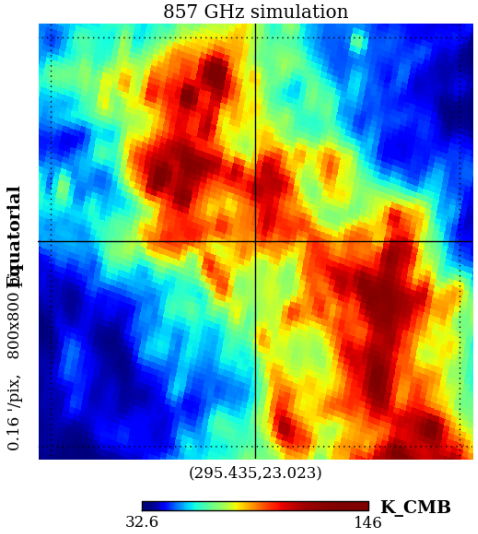


FFP6 Nominal Submm T Maps









CMB Data Analysis

BUT map-making is only one step in the analysis:

1. Pre-processing (calibration, deglitching, flagging, etc)
2. Noise estimation
3. Map-making
4. Component separation
5. Power spectrum estimation
6. Parameter estimation

typically with iteration after each step.

Power Spectrum Estimation

- Decompose the sky into spherical harmonics and sum over m-modes for each l-mode … but
  - Cut sky
  - Inhomogeneous noise
- Maximum likelihood methods:
  - Iterative maximization of Gaussian likelihood
  - Requires full pixel-pixel noise covariance matrix
  - Scales as $\mathcal{N}_p^3$ cycles, $\mathcal{N}_p^2$ memory, $\mathcal{N}_p^2$ disk
- Monte Carlo methods:
  - De-biasing of pseudo spectrum with transfer function
  - Requires Monte Carlo set of map-realizations
  - Scales as $\mathcal{N}_t$ cycles, $\mathcal{N}_p$ memory, $\mathcal{N}_p$ disk

CMB Datasets

- Fainter signals (smaller scale, polarization) require larger datasets to achieve necessary signal-to-noise

Experiment	Start Date	$\mathcal{N}_t$	$\mathcal{N}_p$	PS Method
COBE	1989	10^9	10^4	ML
BOOMERanG	2000	10^9	10^6	
WMAP	2001	10^{10}	10^7	MC
Planck	2009	10^{12}	10^9	
PolarBear	2012	10^{13}	10^7	
QUIET-II	2015	10^{14}	10^7	
CMBpol	2020+	10^{15}	10^{10}	

- CMB datasets grow with Moore's Law!

Monte Carlo Methods

For a given set of detectors and mission interval:

- CMB Monte Carlos:
  - Calculate effective beam
  - Convolve CMB map realizations with effective beam

    => One-time TOD operation for all realizations
- Noise Monte Carlos:
  - Generate a realization of the noise timestreams
  - Map it

    => TOD operations (simulation, mapping)
    for every realization

The Computational Challenge

- Cost of generating a noise Monte Carlo set:

Number of realizations x

(Cost per simulation + Cost per map-making)

=

$$\mathcal{N}_r \times (\mathcal{N}_t + \mathcal{N}_i \times \mathcal{N}_t) \sim \mathcal{N}_r \times \mathcal{N}_i \times \mathcal{N}_t$$

- For Planck:
  - $10^4 \times 10^2 \times 10^{12} = 10^{18}$ flops
  - At 10% efficiency on 1GHz CPU => 10^5 CPU-days!
- Massive parallelism is essential
  - Only 1 day on 100,000 cores ☺

Achieving Efficiency At Scale

- Supercomputers have a hierarchy of efficiency:
  calculation > communication > input/output
  which gets (exponentially) worse with concurrency.
- Naïve approach:
  - For each realization
    - Simulate detector timestreams (CALC)
    - Write detector timestreams (I/O)
    - Read detector pointings & timestreams (I/O)
    - For each iteration
      - Calculate local pixels (CALC)
      - Reduce distributed map (COMM)
    - Write distributed map (I/O)

SIM

MAP

I/O Optimization

- Remove redundant IO
  - generate simulations on-the-fly
    - Sim + Map => SimMap
  - read common data outside of MC realization loop
- Replace IO with calculation
  - read sparse boresight pointing
  - reconstruct dense detector pointing
- Reduces IO by a factor of:

  realizations x detectors x data/pointing sampling ratio

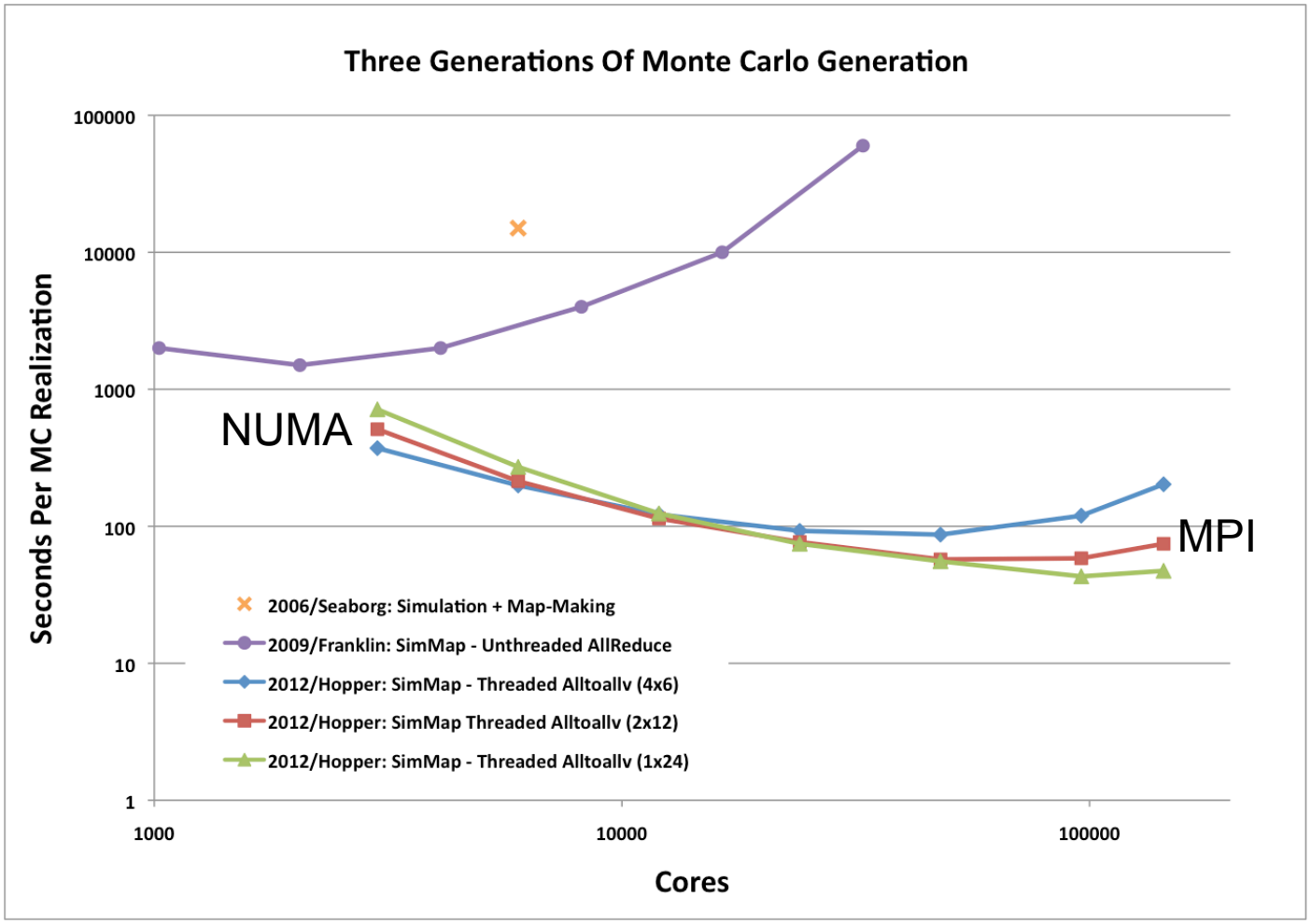
  ~ 10^4 x 10^2 x 10^2 = 10^8 for Planck

COMM Optimization

- Time-ordered data are distributed across cores
  - driven by load-balancing & FFT locality
- Each core has *some* of the samples in *some* of the pixels.
- At each iteration these partial maps must be merged
  - each core needs sum of all samples in pixels it sees

- Reduce concurrency
  - Hybridize: MPI between nodes, threads on nodes
- Reduce communication volume
  - If most cores see most pixels => allreduce
  - If most cores see few pixels => all2allv

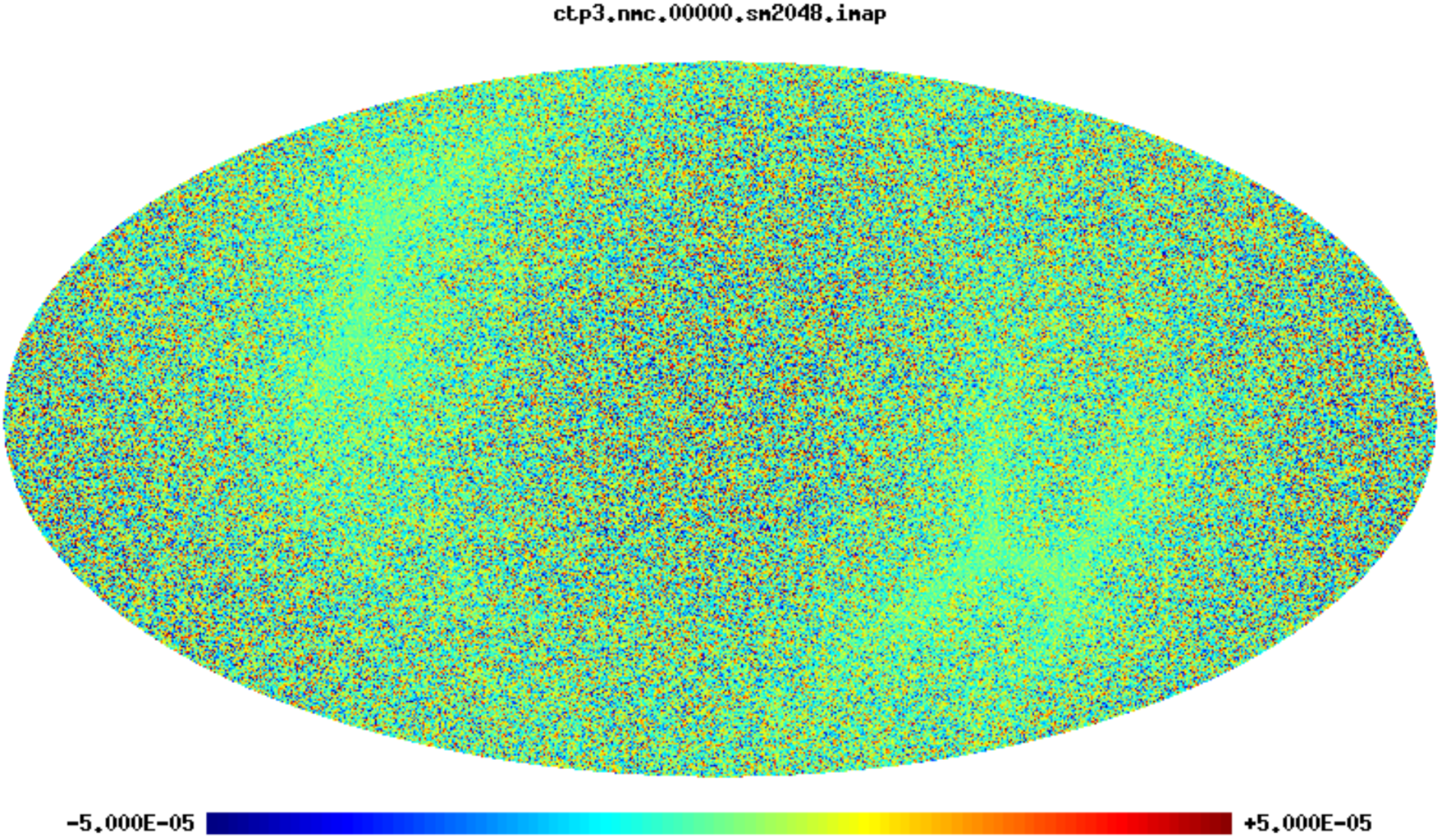
250x Speed-Up



16x Moore's Law (6,000 => 100,000 cores over 6 years)

16x code optimization (break/re-break I/O & comm bottlenecks)

143GHz Nominal Noise MC



Conclusions

- CMB map-making is driven by:
  - Very low signal-to-noise scanning observations
    ⇒ Very large number of samples
  - Very large Monte Carlo requirements
    ⇒ Massively parallel performance is critical
- The situation is only going to get worse!
  - Data volume growing with Moore's Law
  - Polarization requires exquisite control of systematics