Computing Requirements in the 3G Era

Ed Porter
APC / CNRS

Dawn IV, Amsterdam, 30-31 August, 2018

Introduction

- Primer on GW data analysis
- Hardware concerns
- Software concerns
- For the future…
- Conclusion

Disclaimer

- Most of what follows are personal musings!
- There will be NO concrete proposals!!
- There will be many more questions than answers!!!
- I might even get somethings right!!!!

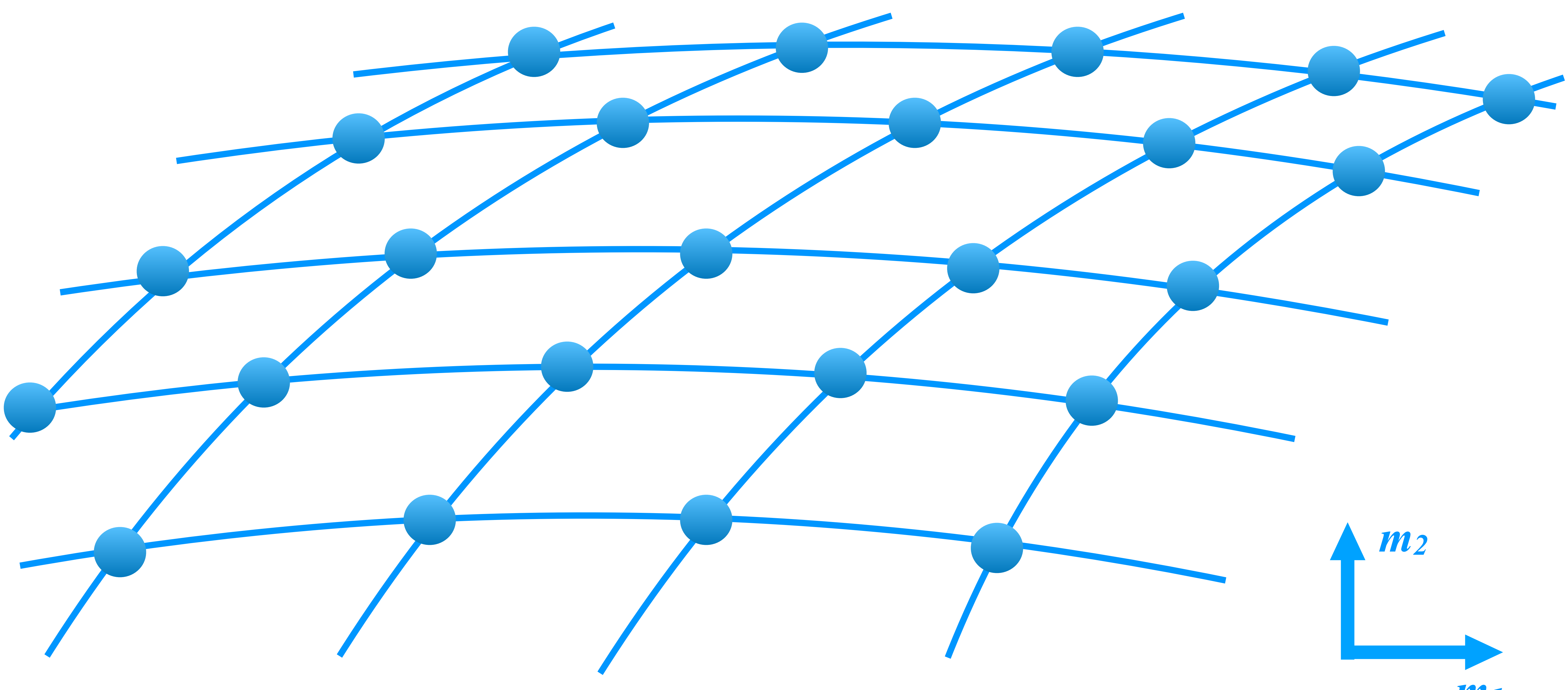
Primer on GW data analysis

Primer on GW data analysis

- Two parts:
  - Detection
  - Parameter estimation (Bayesian inference)

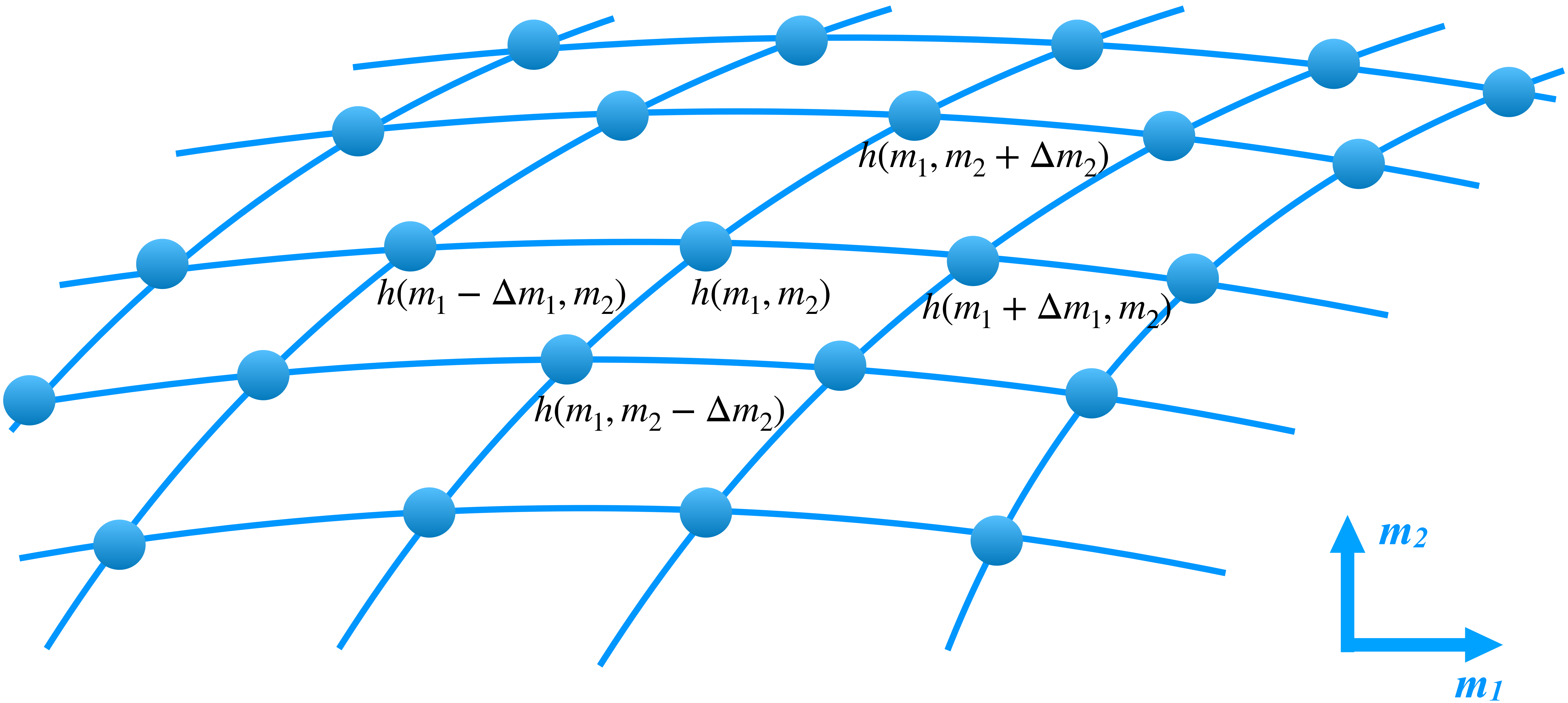
Detection

- Template Grid : assume 2D



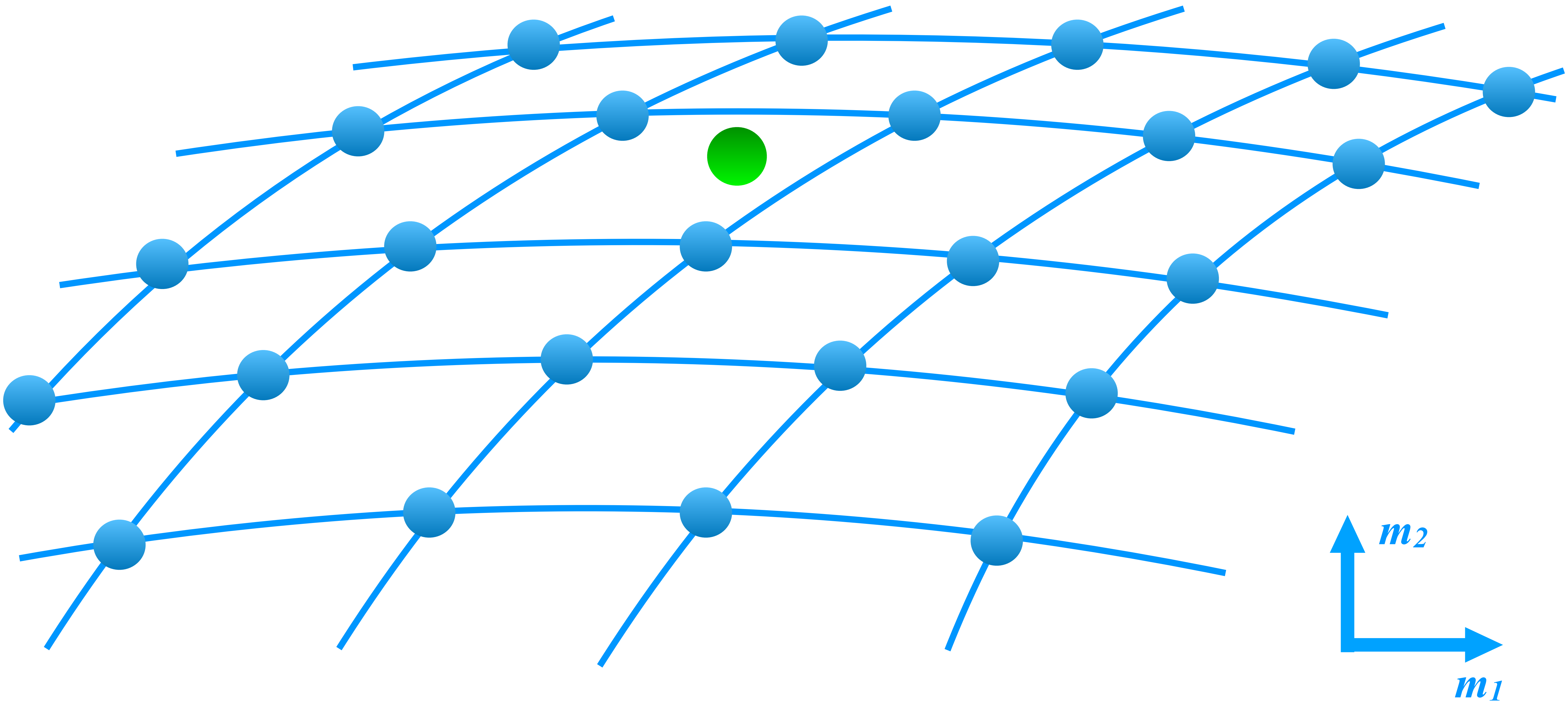
Detection

- Template Grid



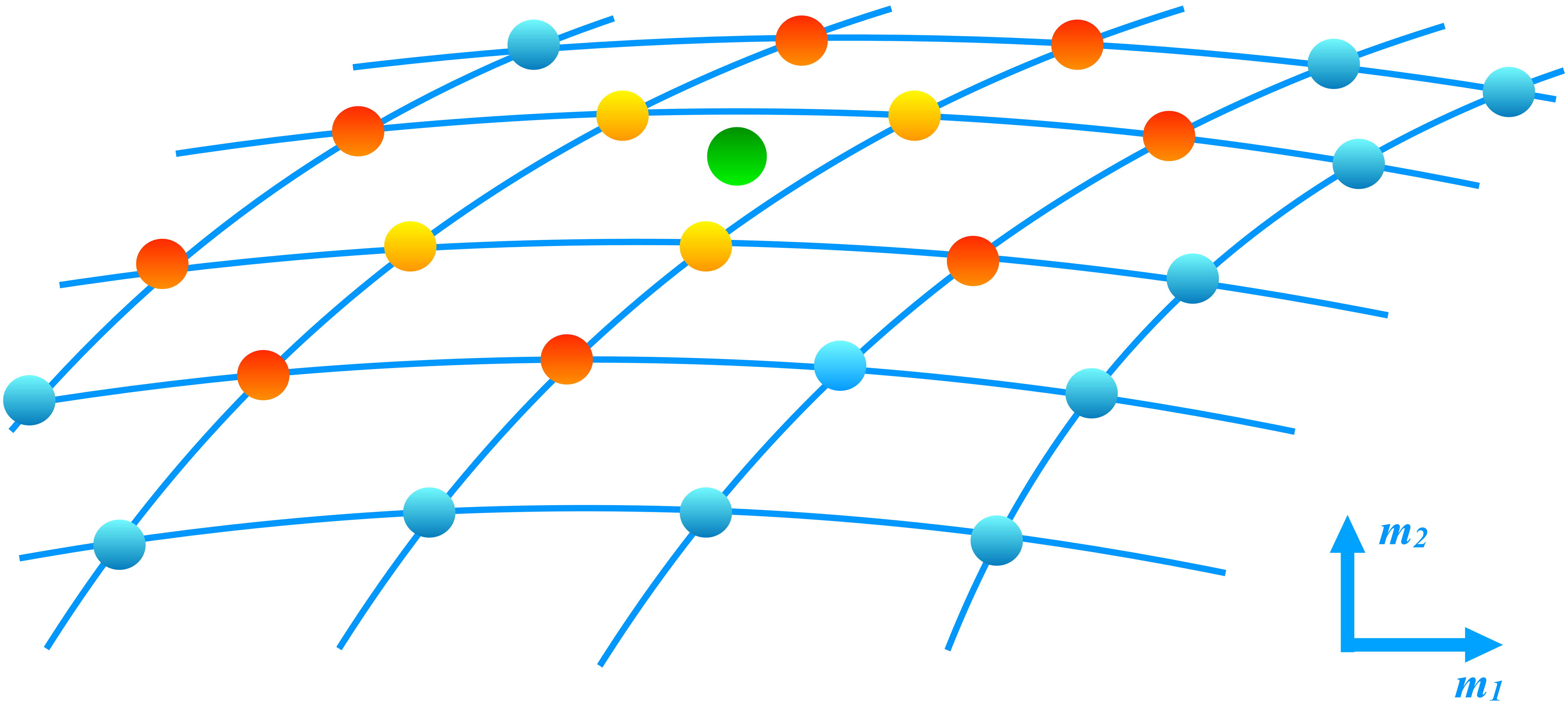
Detection

- Template Grid

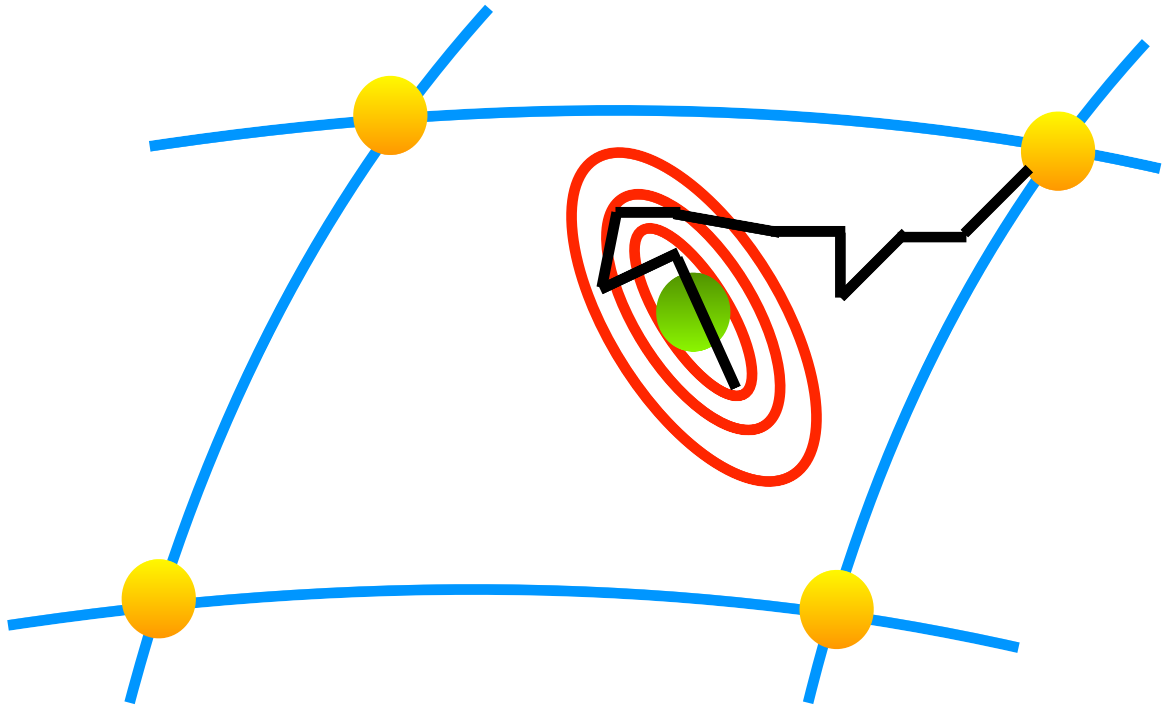


Detection

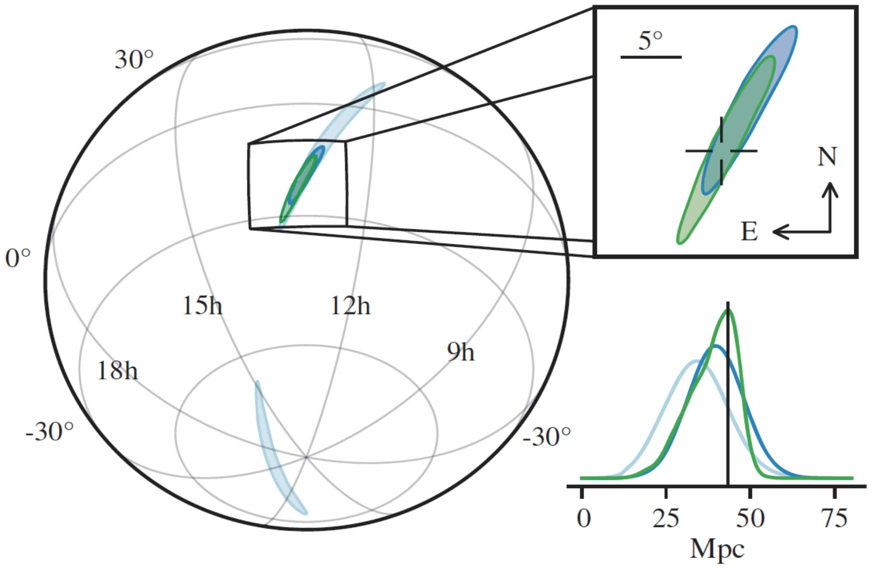
- Template Grid



Bayesian Inference



- Extract astrophysical parameters
- Provide posterior distributions…
- …and confidence intervals



GW170817, B. Abbott et al, PRL 119, 161101 (2017)

Primer on GW data analysis

Detection

- Get the data
- Generate a bank of N templates
- Cross-correlate the templates with the data
- Find template with parameters closest to signal

Bayesian Inference

- Get the data
- Start with parameters of best-match template
- Use a stochastic sampler requiring M templates
- Extract posterior distributions

Primer on GW data analysis

Detection

- Get the data
- Generate a bank of N templates
- Cross-correlate the templates with the data
- Find template with parameters closest to signal

Bayesian Inference

- Get the data
- Start with parameters of best-match template
- Use a stochastic sampler requiring M templates
- Extract posterior distributions

The problem is that N and M are big!!!

Primer on GW data analysis

- **More mathematically**

Detection

$$s(t) = h(t; \vec{\lambda}^*) + n(t)$$

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu$$

$$g_{\mu\nu} = \frac{1}{2}\Gamma_{\mu\nu} = \frac{1}{2} < \partial_\mu h | \partial_\nu h >$$

$$\mathcal{L}(\vec{\lambda}) = \langle s - h | s - h \rangle$$

Bayesian Inference

$$s(t) = h(t; \vec{\lambda}^*) + n(t)$$

$$p(\vec{\lambda}|s) = \frac{\mathcal{L}(\vec{\lambda})\pi(\vec{\lambda})}{p(s)}$$

Primer on GW data analysis

- **More mathematically**

Detection

$$s(t) = h(t; \vec{\lambda}^*) + n(t)$$

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu$$

$$g_{\mu\nu} = \frac{1}{2}\Gamma_{\mu\nu} = \frac{1}{2} < \partial_\mu h | \partial_\nu h >$$

$$\mathcal{L}(\vec{\lambda}) = \langle s - h | s - h \rangle$$

Bayesian Inference

$$s(t) = h(t; \vec{\lambda}^*) + n(t)$$

$$p(\vec{\lambda}|s) = \frac{\mathcal{L}(\vec{\lambda})\pi(\vec{\lambda})}{p(s)}$$

Primer on GW data analysis

- **The noise-weighted inner product**

$$< h|s >= 2\int_{f_{low}}^{f_{high}} \frac{df}{S_n(f)}\tilde{h}(f)\tilde{s}^*(f) + c.c.$$

- **And because we are using computers, the discretized version is**

$$< h|s >= 4\sum_{k=0}^{n/2} h_k s_k^*/S_k$$

- **Each evaluation requires a template generation.**
- **Each search / inference run can require millions - tens of millions of template generations**
- n **can be very large causing both hardware and software problems (more later)**

Primer on GW data analysis

- **The noise-weighted inner product**

$$< h|s >= 2\int_{f_{low}}^{f_{high}} \frac{df}{S_n(f)}\tilde{h}(f)\tilde{s}^*(f) + c.c.$$

- **And because we are using computers, the discretized version is**

$$< h|s >= 4\sum_{k=0}^{n/2} h_k s_k^* / S_k$$

- **Each evaluation requires a template generation.**
- **Each search / inference run can require millions - tens of millions of template generations**
- n **can be very large causing both hardware and software problems (more later)**

So, our question is quite simple…how do we accelerate the generation of templates and the evaluation of the inner product?

Primer on GW data analysis

- The noise-weighted inner product

$$< h|s >= 2\int_{f_{low}}^{f_{high}} \frac{df}{S_n(f)}\tilde{h}(f)\tilde{s}^*(f) + c.c.$$

- And because we are using computers, the discretized version is

$$< h|s >= 4\sum_{k=0}^{n/2} h_k s_k^* / S_k$$

- Each evaluation requires a template generation.
- Each search / inference run can require millions - tens of millions of template generations
- n can be very large causing both hardware and software problems (more later)

Turns out that the solution is not so simple!!!!

Hardware concerns

Solution 1: we just wait...

- So, we just wait for better computers
- I mean, computers in the 3G era will be awesome!!

Moore’s Law

- Gordon Moore (Intel, 1965)
- Number of components on an integrated circuit would double every year
- Revised in 1975 to every two years
- Intel (2015) : 22nm ⇒ 14nm (2.5 years)

  14nm ⇒ 10nm (3 years)

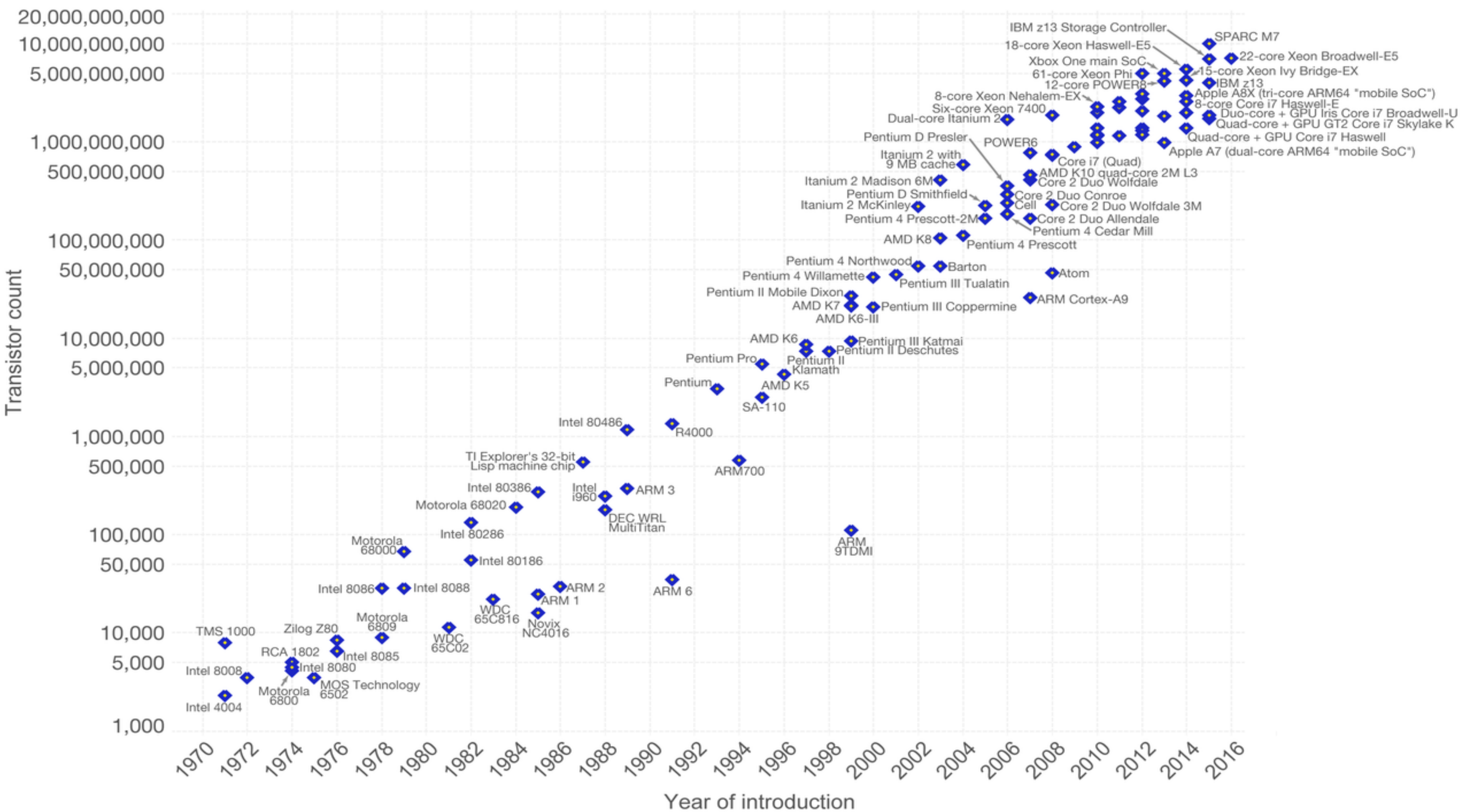
- Believed that Moore’s law will end around 2025 (G. Moore, 2015)

Moore's Law

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Our World in Data

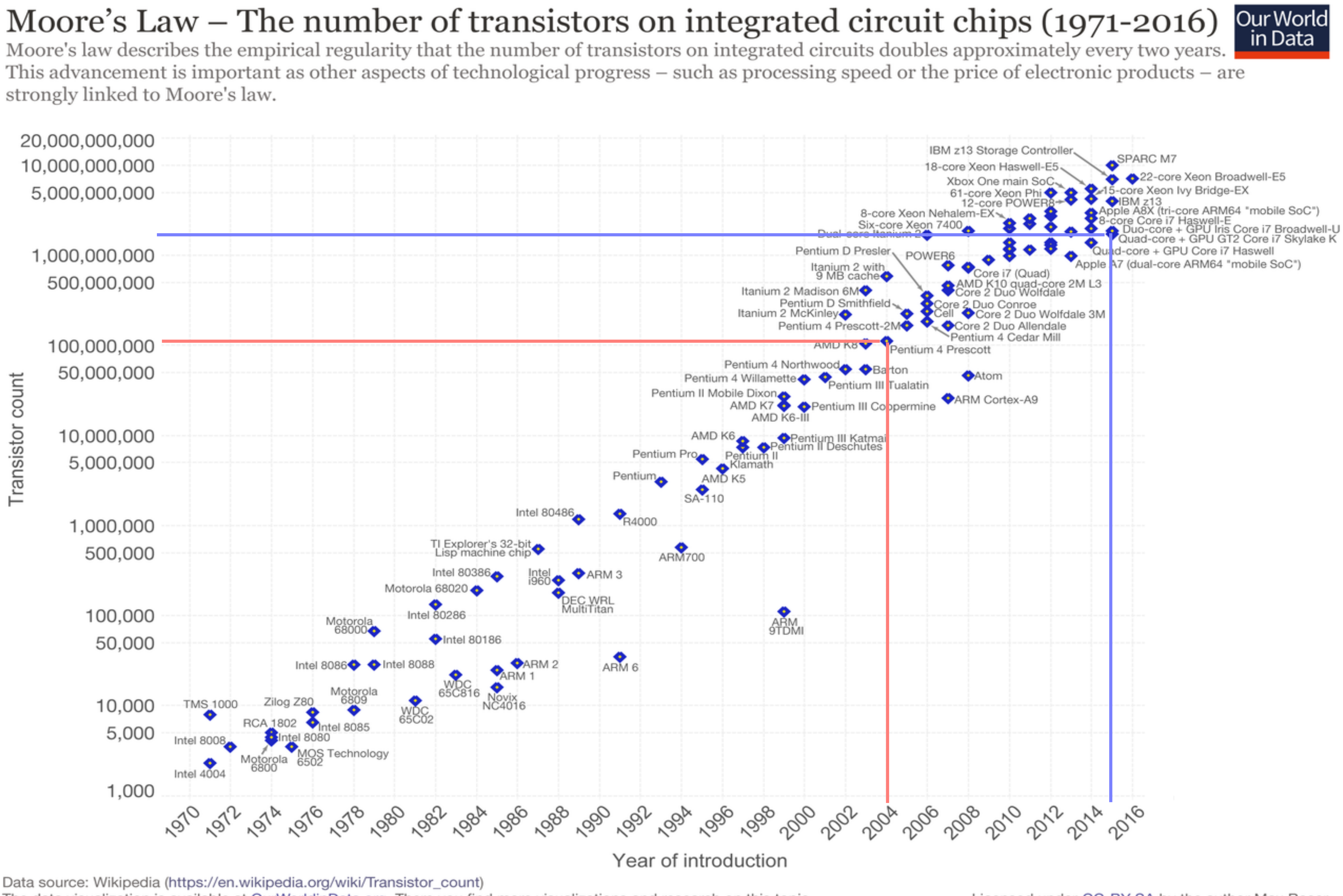
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Moore's Law



Moore's Law – The number of transistors on integrated circuit chips (1971-2016)
Our World in Data
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.
Transistor count
20,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
1970 1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016
Year of introduction
IBM z13 Storage Controller
SPARC M7
18-core Xeon Haswell-E5
22-core Xeon Broadwell-E5
Xbox One main SoC
15-core Xeon Ivy Bridge-EX
61-core Xeon Phi
IBM z13
12-core POWER8
Apple A8X (tri-core ARM64 "mobile SoC")
8-core Xeon Nehalem-EX
8-core Core i7 Haswell-E
Six-core Xeon 7400
Duo-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Quad-core + GPU Core i7 Haswell
Pentium D Presler
POWER6
Apple A7 (dual-core ARM64 "mobile SoC")
Itanium 2 with 9 MB cache
Core i7 (Quad)
AMD K10 quad-core 2M L3
Itanium 2 Madison 6M
Core 2 Duo Wolfdale
Pentium D Smithfield
Core 2 Duo Conroe
Itanium 2 McKinley
Cell
Core 2 Duo Wolfdale 3M
Pentium 4 Prescott-2M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
AMD K8
Pentium 4 Prescott
Pentium 4 Northwood
Barton
Pentium 4 Willamette
Atom
Pentium III Tualatin
Pentium II Mobile Dixon
AMD K7
Pentium III Coppermine
ARM Cortex-A9
AMD K6-III
AMD K6
Pentium III Katmai
Pentium II Deschutes
Pentium Pro
Pentium II Klamath
Pentium
AMD K5
SA-110
Intel 80486
R4000
TI Explorer's 32-bit Lisp machine chip
ARM700
Intel 80386
Intel i960
ARM 3
Motorola 68020
DEC WRL MultiTitan
Intel 80286
ARM 9TDMI
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
WDC 65C816
ARM 1
ARM 6
Motorola 6809
Novix NC4016
TMS 1000
Zilog Z80
WDC 65C02
RCA 1802
Intel 8085
Intel 8080
Intel 8008
MOS Technology 6502
Motorola 6800
Intel 4004
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.
Licensed under CC-BY-SA by the author Max Roser.

Moore's Law

- 2006 : Bayesian inference of SMBHB for LISA (Cornish & Porter, CQG23, S761 (2006)). 10^7-$10^6\ M_\odot$ @ $z = 1$. $t_c = 0.49$ yr, $T_{obs} = 0.5$ yr. $n = 16384 = 2^{14}$. Runtime for a $8\mathrm{x}10^6$ MCMC on a Dell desktop with a 2GHz Pentium 4 processor: 8 days

Moore's Law

- 2006 : Bayesian inference of SMBHB for LISA (Cornish & Porter, CQG23, S761 (2006)). 10^7-10^6 $M_\odot$ @ $z = 1$. $t_c = 0.49$ yr, $T_{obs} = 0.5$ yr. $n = 16384 = 2^{14}$. Runtime for a $8\text{x}10^6$ MCMC on a Dell desktop with a 2GHz Pentium 4 processor: 8 days

- 2018 : Same source. Runtime on a 2017 13" Macbook Pro with a 2.9 GHz i5 processor: 44 hours

- Jumping from $100\text{x}10^6$ to $2\text{x}10^9$ transistors gives a speed-up of x4.4??

- Not very impressive...so what's going on???

Wirth's/Gate's/May's Law

- Also known as "software bloat"!!
- Successive generations of software offset the gains given by Moore's law: commercial software slows by 50% every 18 months
- Office 2007 performed the same task on a typical year-2007 computer at half the speed of Office 2000 on a typical year-2000 computer (Randall Kennedy, Intel, 2008)
- Intel/AMD now prefer multicore chips using multi-threading. As a consequence, the software has to be written in a multi-threaded manner to take advantage of the hardware
- MCMC is a sequential algorithm that uses a single thread…hence the reason for the modest improvement.
- So, to take advantage of multithread technology, I need to write my MCMC code in a multithread fashion…..except….I won't!!!!

Physics...

- As we approach 5nm, physics becomes a problem
- gate design using Si is an issue at this level
- Research has started into alternative materials, e.g. InGaAs, Graphene
- No current estimates on success!!

Economics…

- The cost of a silicon chip fabrication plant doubles in price every 4 years
- Improvements will only continue as long as profit > cost

Sofware concerns

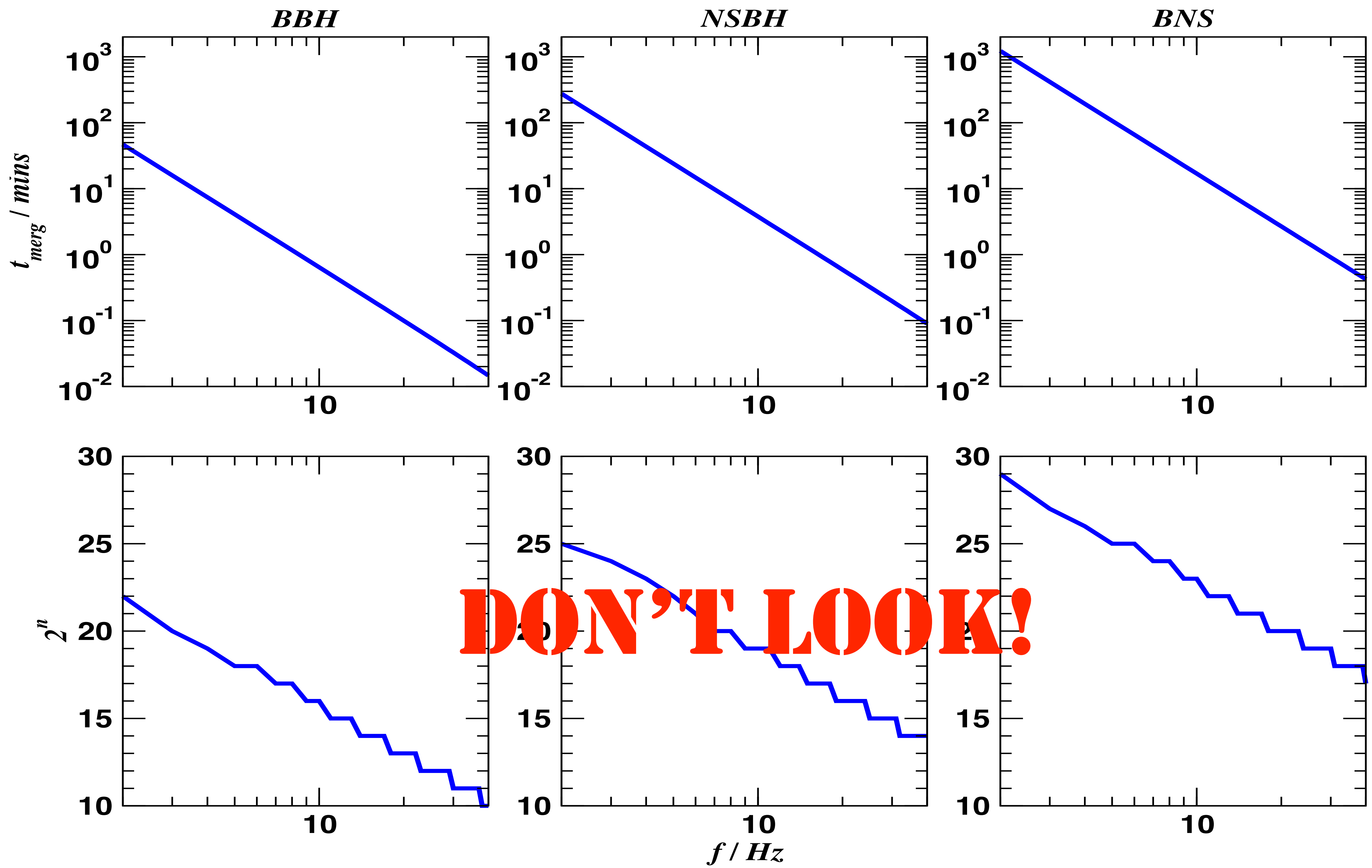
Template Banks

- The big question is will be able to continue using template banks in 3G?
- Different problem, but, for LISA we demonstrated that a template bank search would require 10^{12} templates for SMBHBs (Cornish & Porter, 2005), 10^{7}-10^{11} templates for GBs (Cornish and Porter, 2005) and 10^{40} templates for EMRIs (Gair et al, 2004)
- 3G ground-based template banks will be smaller, but will still be big!
- Given that it is possible that there will be source confusion, and that template banks will be to costly, more stochastic search methods may be needed (Bosi & Porter, 2011)
- But, let's imagine we can use template banks….

Template Length

- Assume we are are going to target BBH, BNS and NSBH systems separately
- Assume a minimum mass of 1 $M_\odot$ for a NS and 5 $M_\odot$ for a BH
- Assume the longest template in any search/PE is equal mass
- Use a 3.5PN chirp time to calculate template duration
- Assume each template has a sampling frequency of $2f_{Nyq}$, where the Nyquist frequency assumes minimum total masses of 2, 6 and 10 $M_\odot$ for each search

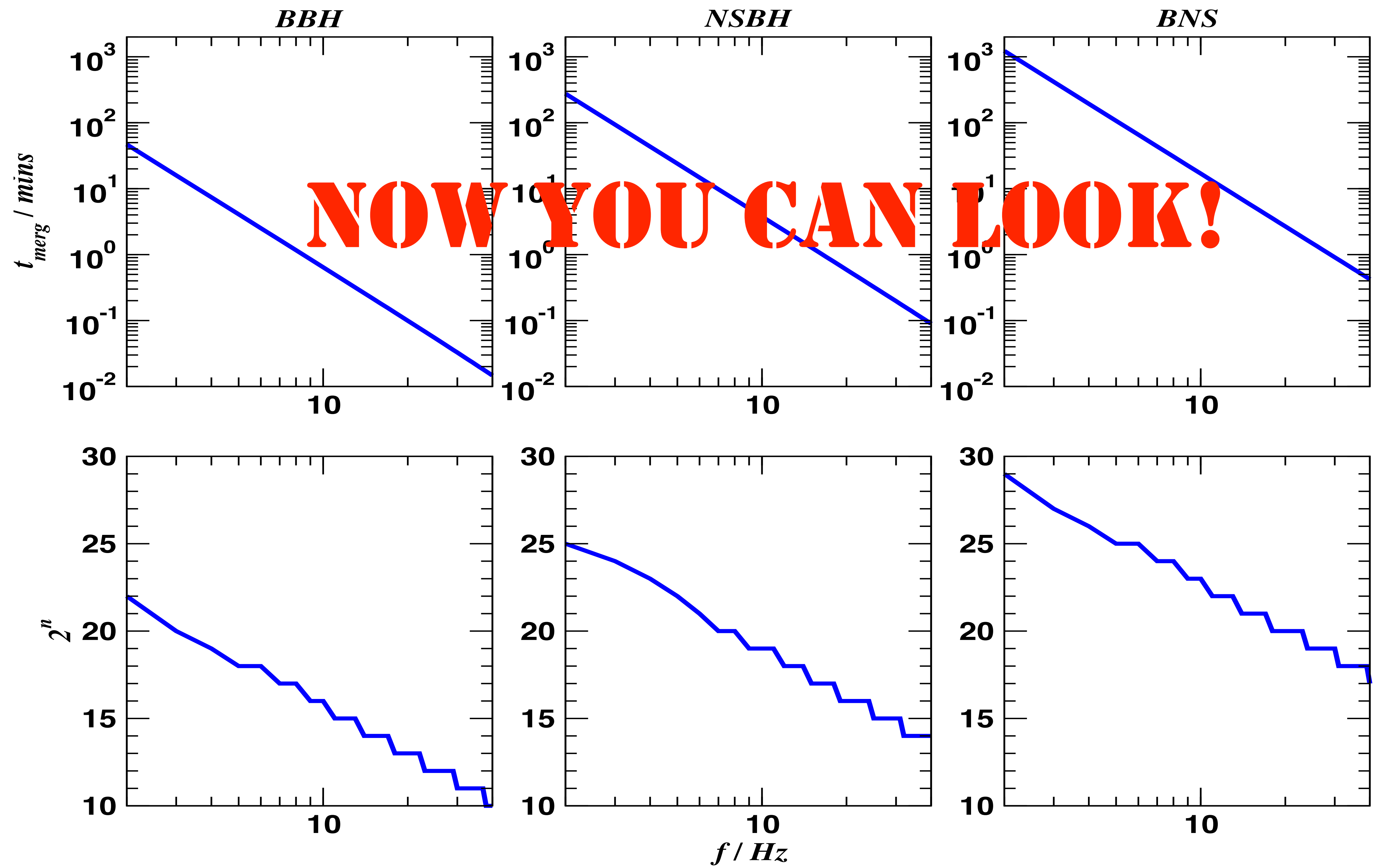
Template Duration



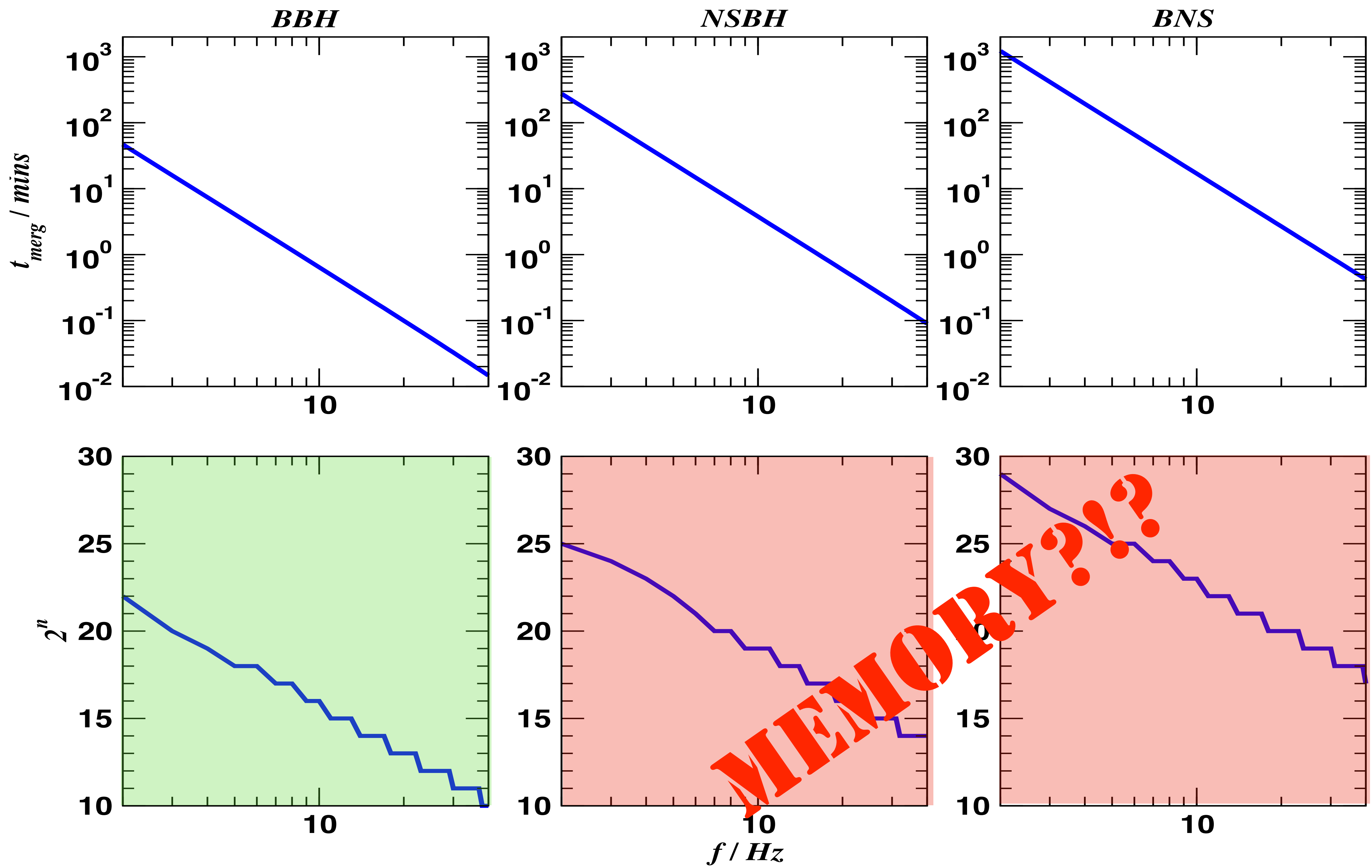
Template Duration

f / Hz	BBH	NSBH	BNS
30	2 secs	12 secs	1 min
20	6 secs	36 secs	3 min
15	13 secs	76 secs	6 min
10	40 secs	4 min	17 min
9	50 secs	5 min	22 min
8	70 secs	7 min	30 min
7	1.5 min	10 min	45 min
6	2.5 min	15 min	1 hr
5	4 min	25 min	1.75 hr
4	8 min	45 min	3 hr
3	16 min	1.5 hr	7 hr
2	45 min	5 hr	20 hr

Template Duration



Template Duration



Computational Power

- To get an idea of the necessary computational power (Schutz, 1989)
- To filter N templates of length F through the data, where F is given by

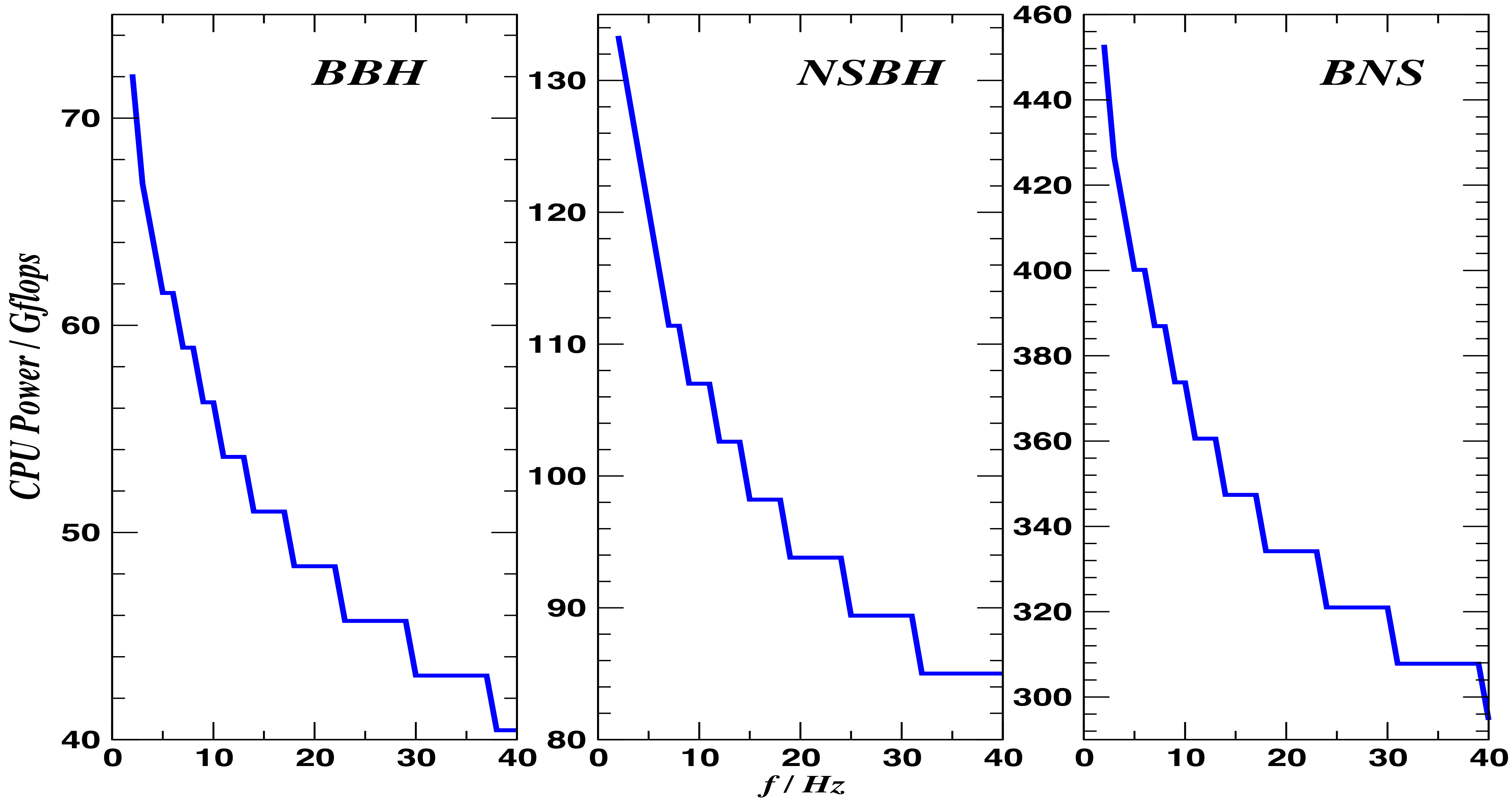
$$F = \frac{5}{32} f_{Nyq} \left(\pi f_{low}\right)^{-8/3} \left(2GM_{min}/c^3\right)^{-5/3}$$

- requires a computational power, in flops, of

$$P \approx N f_{Nyq} \left(32 + 6 \log_2 F\right)$$

- assuming M_{min} has η= 1/4, and $f_{samp} = 2\, f_{Nyq}$
- Assume my template bank requires 10^6 templates

Template Duration

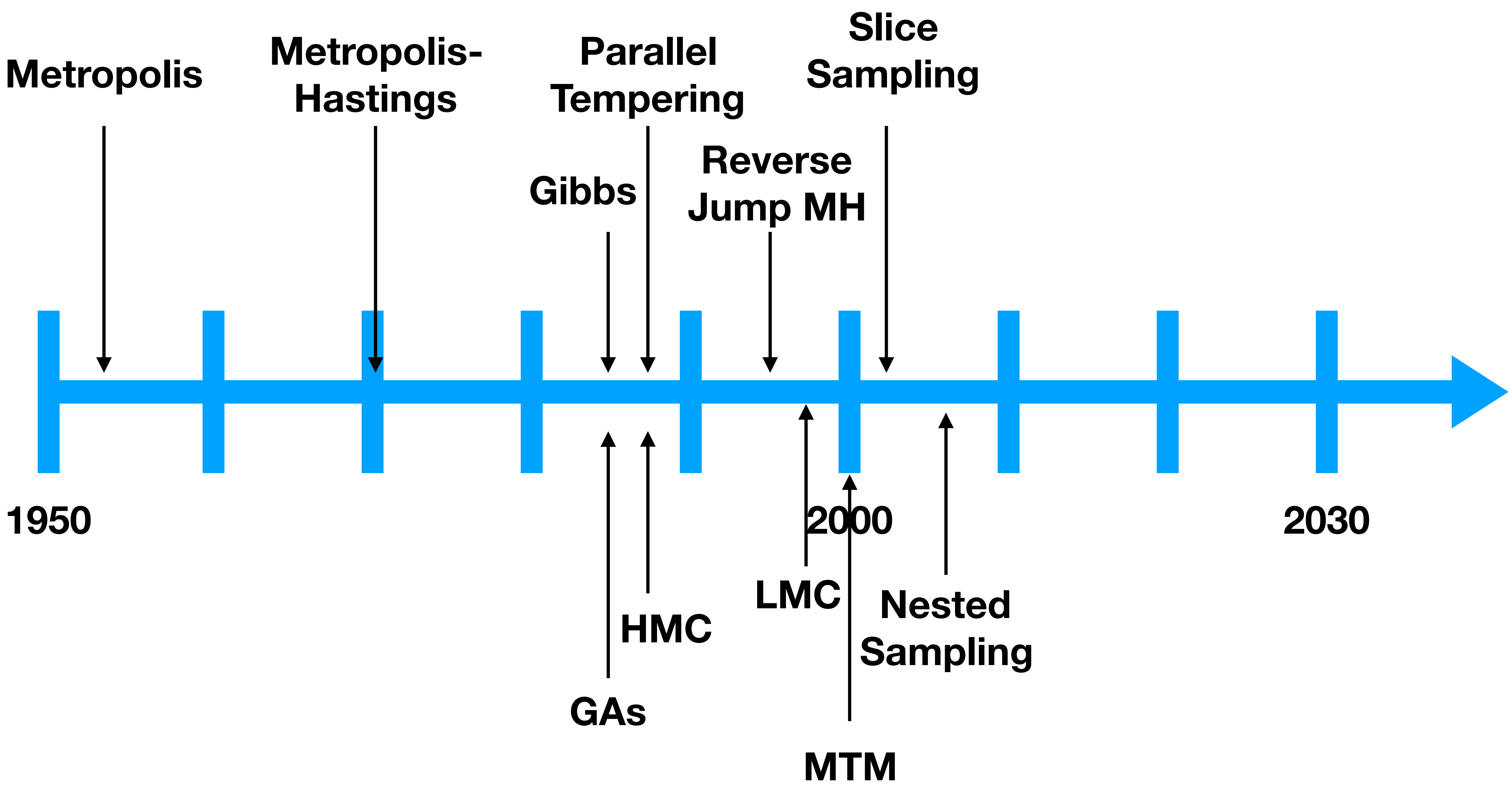


N.B. : CPU power scales linearly with template number

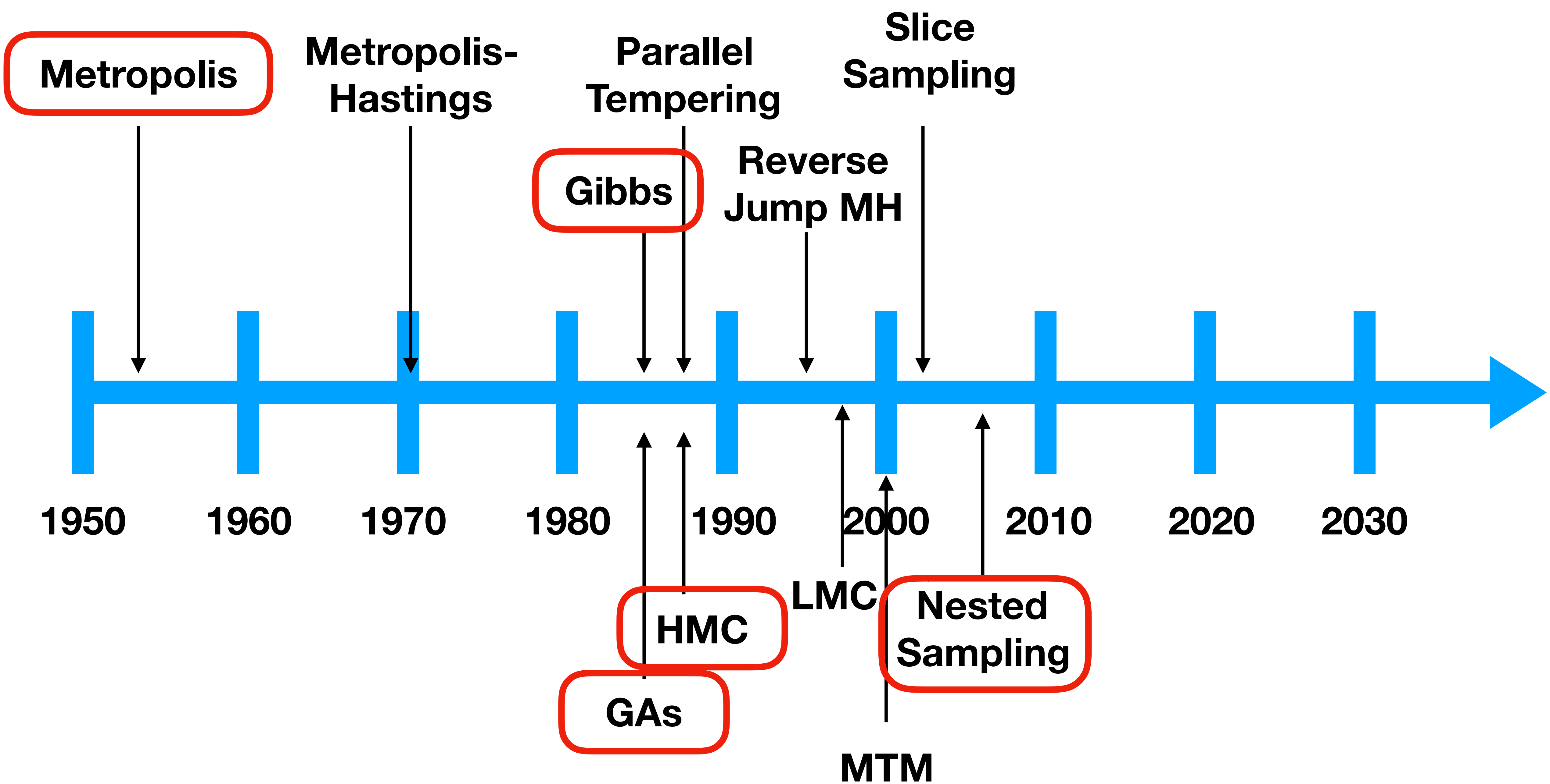
Solution 2:

- We just use better algorithms
- There must be many to choose from..
- Bayesian inference is used in so many fields, e.g. economics, air traffic control, biology, astronomy, computer chess, computational music etc
- I mean, let's face it, everyone is a Bayesian these days!!!
- Well……

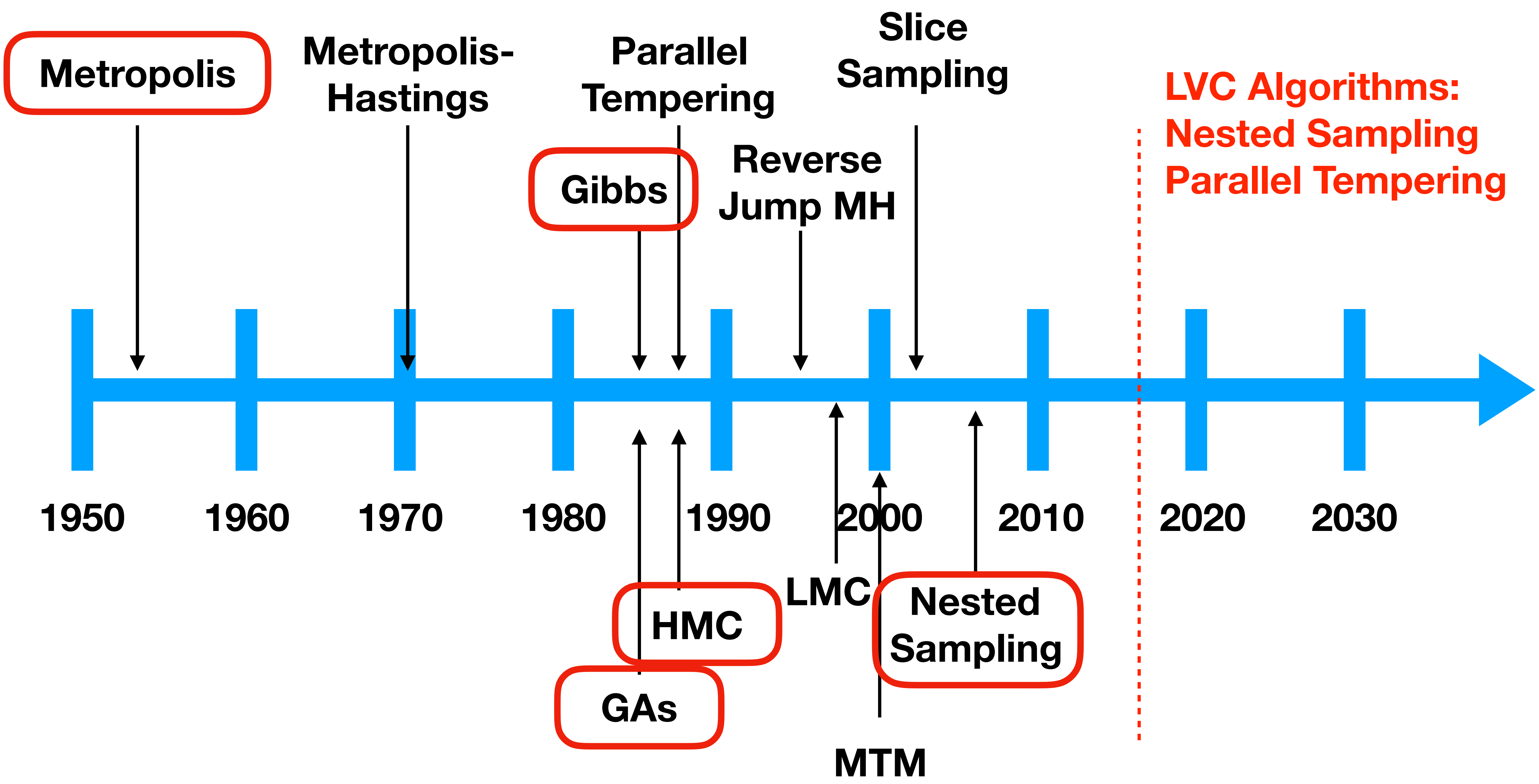
Bayesian Inference Algorithms



Bayesian Inference Algorithms



Bayesian Inference Algorithms



Bayesian Inference Algorithms

- Development of a new algorithm is not trivial
- In most cases, “off the shelf” algorithms do not work for your problem
- That means the development of GW-specific algorithms
- Have to solve efficiency and convergence issues
- Difficult to do while analysing data
- Even today, we require 10s of millions of likelihood evaluations to achieve an acceptable number of SISs

For the future...

Biggest advice....

- Look before you jump….

Remember…the PS3??

- Released in 2006-7 with a 6/8 core processor
- You could install Linux/Unix
- It was going to revolutionise scientific computing
- We were all going to build PS3 clusters…
- …except you couldn't keep them cool enough…
- …and in 2010, citing security concerns, Sony removed the ability to install other OSs

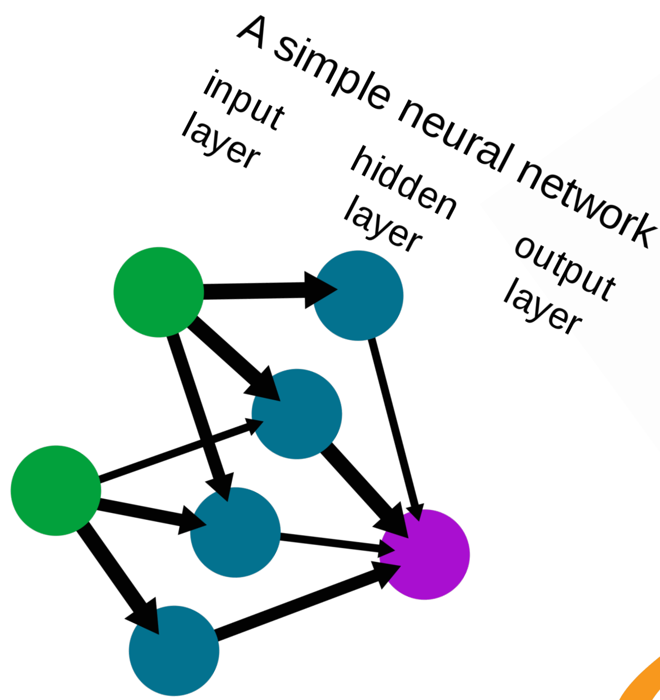
or the GPU??

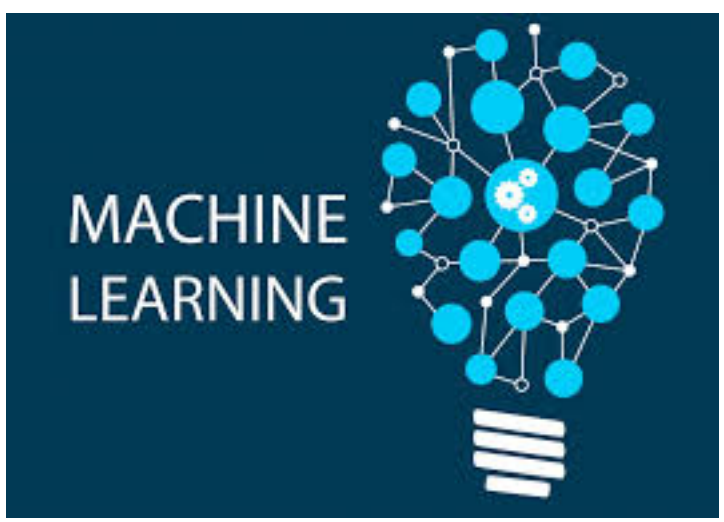
- Who needs a 2-4 core CPU, when I can have a 512-1024 core GPU
- It was going to revolutionise scientific computing
- We were all going to build GPU clusters
- Except you don't really know which language to use…
- You have to solve the IO problem…
- To get maximum benefit, you have to program to the hardware requirements…

But we can use the cloud!!

- Don't need to invest in hardware
- Can build virtual clusters
- Can use the "cloudburst" feature
- It'll revolutionise scientific computing!!!
- Big question : will the cloud still be there in the 3G era?
- From "talking with people"…doubtful!! Will be replaced with something else in the next decade!!

But, things are easier with software…





On a serious note...

- Hardware:
  - Agency/University clusters
  - GRID
  - Laptops/Desktops
  - task force investigating new technology?
  - Cost/benefit analysis?
- Software:
  - Do we need to move beyond scientist developing codes?
  - Computer scientists / professional programmers for profiling / optimisation
  - Investigation and benchmarking of new methods (+ development)
  - Cost/benefit analysis?
  - We will still use externally developed code, e.g. FFTW, astropy, numpy, BLAS, gsl

Conclusion

- GW data analysis will become more difficult and demanding
- Storage will become an issue
- Performance will be an issue depending on how low-latency we want to be
- It may require an investment in computing that frees up the scientists to do science
- We need to keep an eye on emerging technologies…but resist jumping into them too early
- Not everything that's useful, is useful for GW astronomy