FastJet

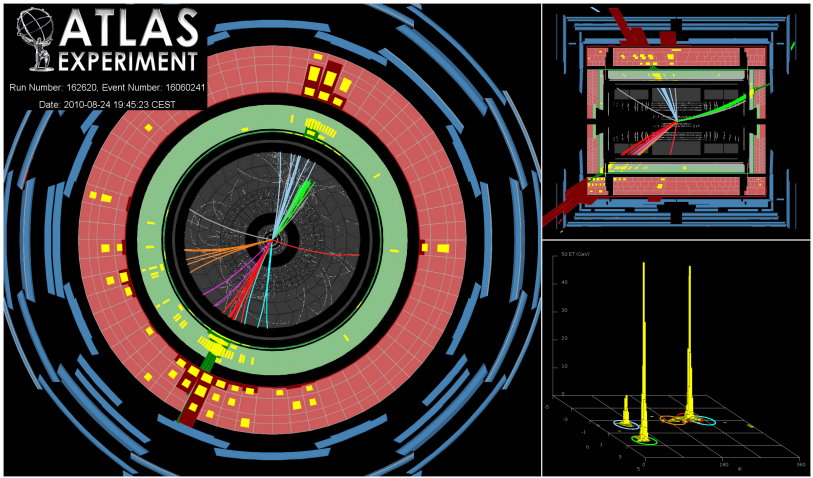
Gregory Soyez (with Matteo Cacciari and Gavin Salam)

IPhT, CNRS, CEA Saclay

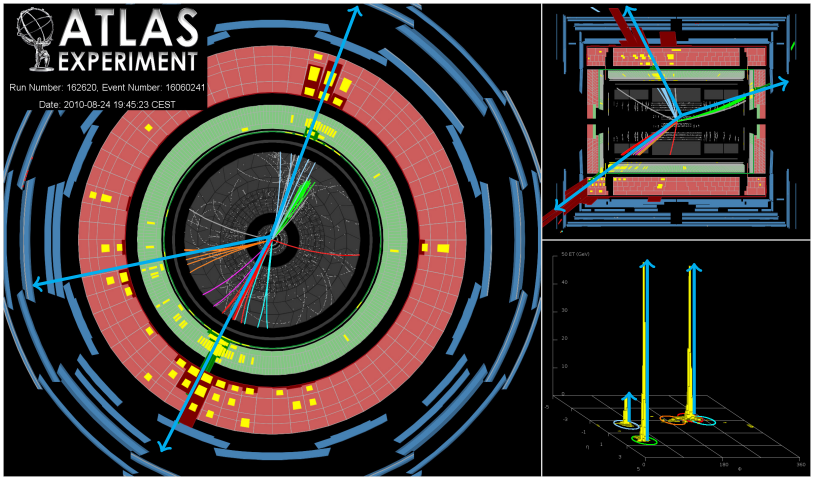
Joint GDR-QCD/Strong2020, May 31-June 4 2021, IJCLab (online)

Outline

- Physics background: jets and clustering
- FastJet: fast clustering
- FastJet: jet manipulations



Particles/Energy flow organised in a few dominant directions $\Rightarrow$ JETS

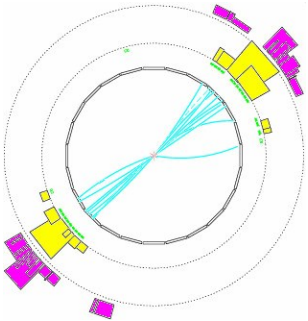


Particles/Energy flow organised in a few dominant directions $\Rightarrow$ JETS

Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

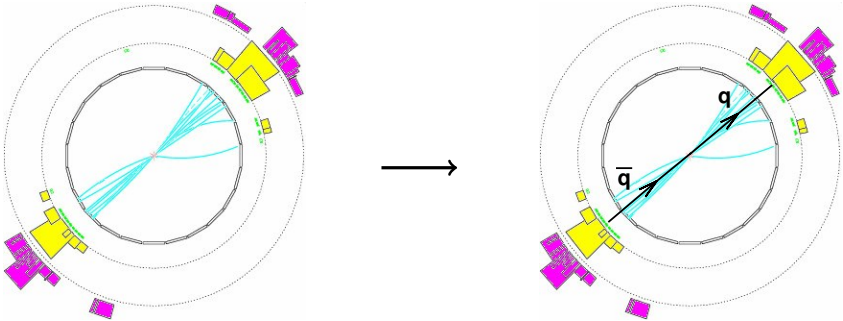
How many jets?



Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

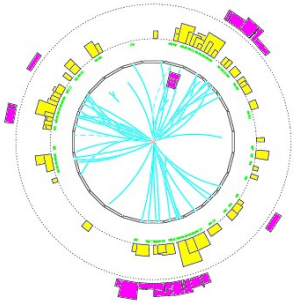
obviously 2 jets



Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

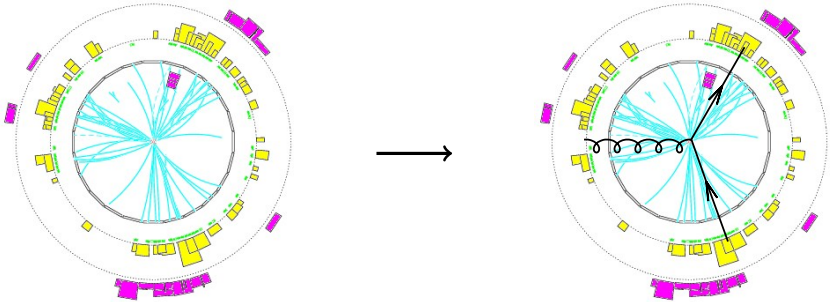
How many jets?



Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

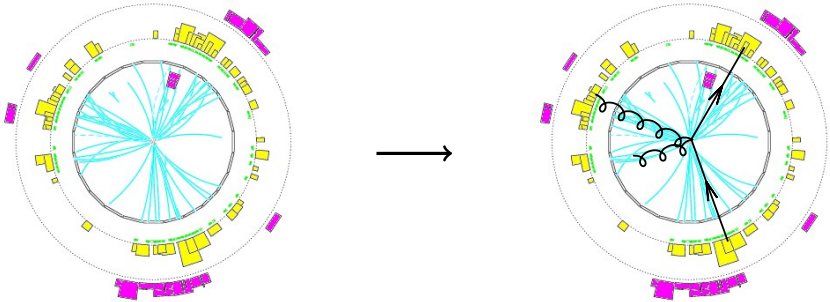
3 jets



Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

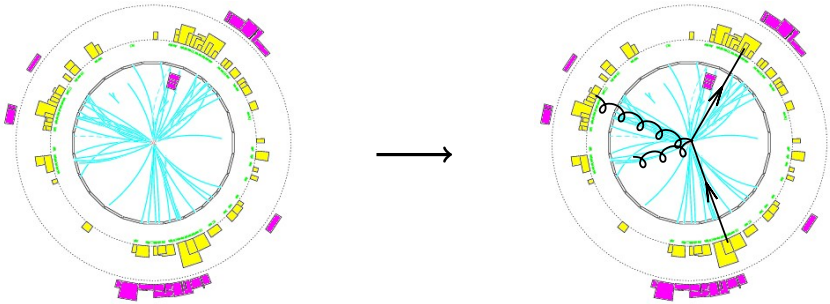
3 jets... or 4?



Jets and partons

"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

3 jets... or 4?

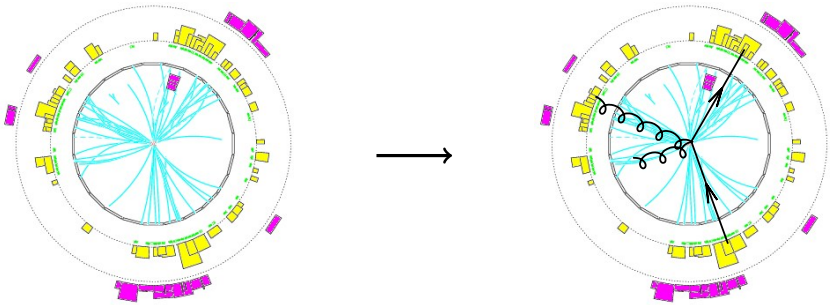


- "collinear" is arbitrary + "parton" concept strictly valid only at LO

Jets and partons

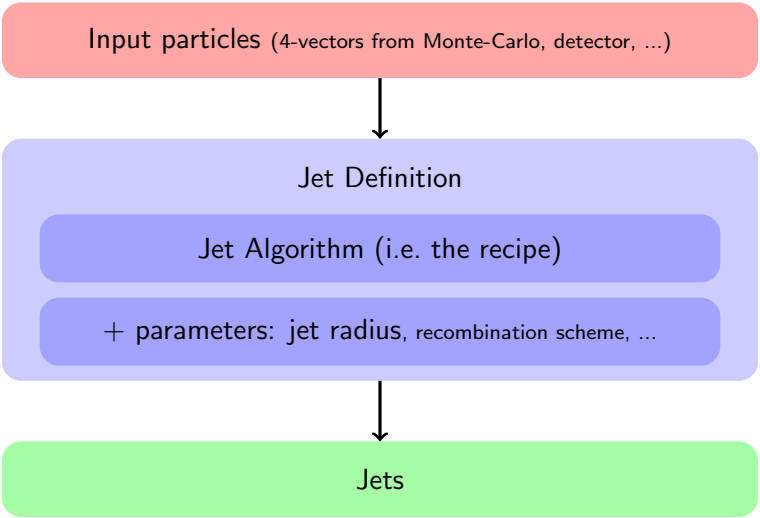
"Jets" $\equiv$ bunch of collimated particles $\cong$ hard partons

3 jets... or 4?

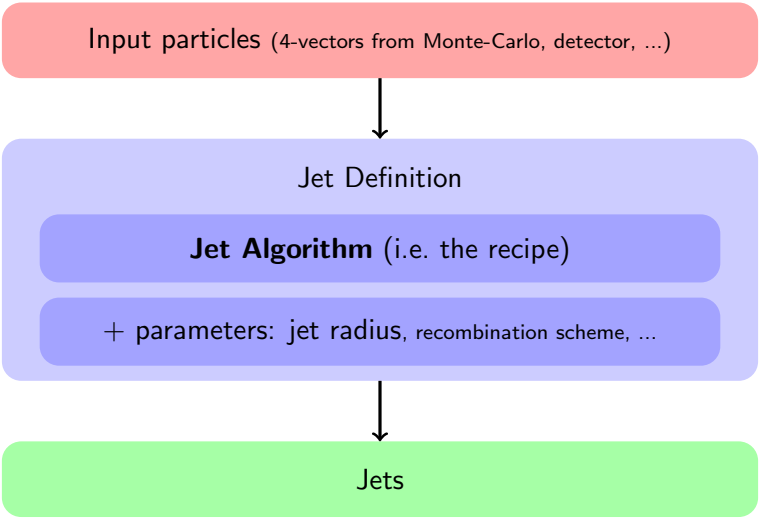


- "collinear" is arbitrary + "parton" concept strictly valid only at LO
- Define jets instead

Jet algorithms and definitions



Jet algorithms and definitions



Most common approach today: recombination algorithms

Generalised-k_t algorithm

- From all the objects to cluster, define the distances

$$d_{ij} = \min(p_{t,i}^{2p}, p_{t,j}^{2p})(\Delta y_{ij}^2 + \Delta\phi_{ij}^2), \qquad d_{iB} = p_{t,i}^{2p} R^2$$

- repeatedly find the minimal distance
  if d_{ij}: recombine i and j into $k = i + j$
  if d_{iB}: call i a jet

Most common approach today: recombination algorithms

Generalised-k_t algorithm

- From all the objects to cluster, define the distances

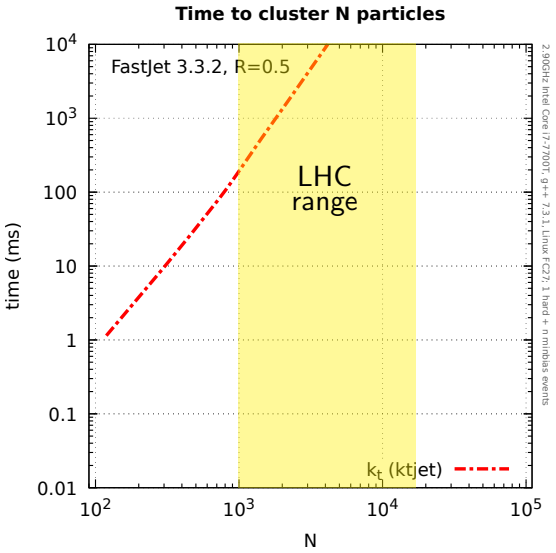
$$d_{ij} = \min(p_{t,i}^{2p}, p_{t,j}^{2p})(\Delta y_{ij}^2 + \Delta\phi_{ij}^2), \qquad d_{iB} = p_{t,i}^{2p} R^2$$

- repeatedly find the minimal distance
  if d_{ij}: recombine i and j into $k = i + j$
  if d_{iB}: call i a jet

- Parameter p is (typically) one of
  - $p = 1$: k_t algorithm (closest to QCD) [Catani,Dokshitzer,Seymour,Weber,Ellis,Soper,1993]
  - $p = 0$: Cambridge/Aachen (geometrical distance) [Dokshitzer,Leder,Moretti,Webber,1997]
  - $p = -1$: anti-k_t (the LHC choice) [M.Cacciari,G.Salam,GS,2008]

FastJet: a fast implementation of (generalised)-k_t

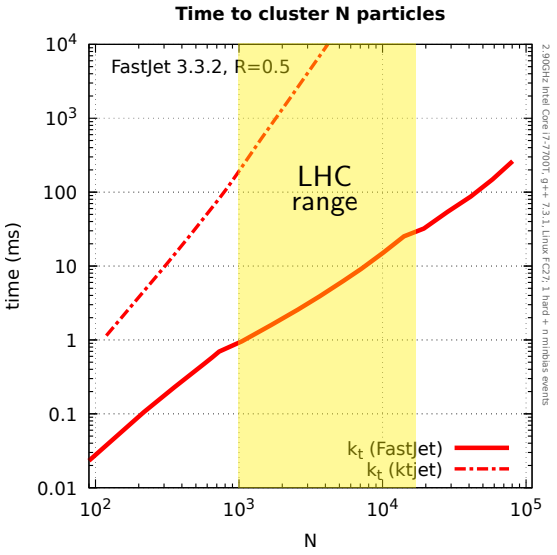
Fast clustering



- k_t Before FastJet
  Complexity $\propto N^3$ (*)

(*) more on that later if enough time

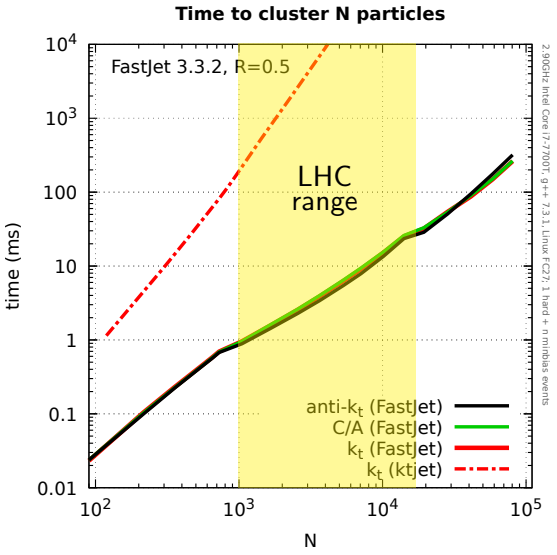
Fast clustering



- k_t Before FastJet
  Complexity $\propto N^3$ (*)
- k_t Fastjet's implementation gain 2-3 orders of magnitude
  Makes if usable at the LHC
  (Trigger spends $\sim$ 100ms/event)
  Complexity $\propto N^2$ or $N \ln N$ (*)

(*) more on that later if enough time

Fast clustering



- k_t Before FastJet
  Complexity $\propto N^3$ (*)
- k_t Fastjet's implementation gain 2-3 orders of magnitude
  Makes if usable at the LHC
  (Trigger spends $\sim$ 100ms/event)
  Complexity $\propto N^2$ or $N \ln N$ (*)
- Similar time for anti-k_t and C/A

(*) more on that later if enough time

Simple code

All done in a few lines of code:

```
// list of input particles/objects
vector<PseudoJet> particles;

// Cluster with anti-k_t, R = 0.5
JetDefinition jet_def(antikt_algorithm, 0.5);

// Get the jets with p_t > 50 GeV, |y| < 2.5
Selector jet_selector = SelectorPtMin(50.0) * SelectorAbsRapMax(2.5);
vector<PseudoJet> jets = jet_selector(jet_def(particles));

// Simple manipulation
for (auto &jet :  jets){
double pt = jet.pt(); // jet pt
vector<PseudoJet> constituents = jet.constituents(); // particles that made
up the jet
```

Simple code

All done in a few lines of code:

```
// list of input particles/objects
vector<PseudoJet> particles;

// Cluster with anti-k_t, R = 0.5
JetDefinition jet_def(antikt_algorithm, 0.5);

// Get the jets with p_t > 50 GeV, |y| < 2.5
Selector jet_selector = SelectorPtMin(50.0) * SelectorAbsRapMax(2.5);
vector<PseudoJet> jets = jet_selector(jet_def(particles));

// Simple manipulation
for (auto &jet :  jets){
double pt = jet.pt(); // jet pt
vector<PseudoJet> constituents = jet.constituents(); // particles that made
up the jet
```

Grown into framework for jet manipulation

(Area-based) background subtraction (pileup or heavy-ion)

Core interface for substructure tools

Discussion about complexity