Online and Offline Pattern Recognition in PANDA

Gianluigi **Boca** for the PANDA Collaboration[1,a]

[1] *Dipartimento di Fisica and INFN, Pavia, Italy*

Abstract. PANDA is one of the four experiments that will run at the new facility FAIR that is being built in Darmstadt, Germany. It is a fixed target experiment: a beam of antiprotons collides on a jet proton target (the maximum center of mass energy is 5.46 GeV). The interaction rate at the startup will be 2 MHz with the goal of reaching 20 MHz at full luminosity. The beam of antiprotons will be essentially continuous. PANDA will have NO hardware trigger but only a software trigger, to allow for maximum flexibility in the physics program. All those characteristics are severe challenges for the reconstruction code that 1) must be fast, since it has to be validated up to 20 MHz interaction rate; 2) must be able to reject fake tracks caused by the remnant hits, belonging to previous or later events in some slow detectors, for example the straw tubes in the central region. The Pattern Recognition (PR) of PANDA will have to run both online to achieve a first fast selection, and offline, at lower rate, for a more refined selection. In PANDA the PR code is continuously evolving; this contribution shows the present status. I will give an overview of three examples of PR following different strategies and/or implemented on different hardware (FPGA, GPUs, CPUs) and, when available, I will report the performances.

1 Introduction

PANDA[1] is an experiment that will run at the FAIR accelerator facility in Darmstadt, Germany, starting in 2021. A beam of antiprotons circulating in an accumulation ring with momentum ranging from 1.5 to 15 GeV/c will collide on a hydrogen jet target. The interaction rate at the beginning will be 2 MHz with the goal of reaching 20 MHz when the accelerator delivers the full luminosity.

The PANDA event selection will be completely software. This has been designed to achieve the maximum possible flexibility in the physics channels selection and to exploit the detector possibilities at the fullest. On the other hand this choice poses a great challenge on the Pattern Recognition (PR) software that must keep up with the initial 4 MHz (corresponding to a raw data rate of 20 GB/s) and, in perspective, with the final 20 MHz interaction rate (raw data rate of 200 GB/s), and it must reject the fake tracks produced by the pile-up of hits from previous or subsequent events. A lot of PR code have been developed so far in PANDA. It can be divided essentially in three categories: 1) online PR running on FPGA ; 2) code running on GPU for online/offline PR; 3) code running on CPU for offline PR.

For lack of space only three examples of PR code are described in the following, one running on FPGAs, one for GPUs and one running on CPUs. At the end a short summary of the PR algorithms developed in PANDA can be found.

[a] e-mail: gianluigi.boca@pv.infn.it

First a very brief description of the PANDA central detector is given in the next section in order to allow the reader to understand later the PR algorithms described.

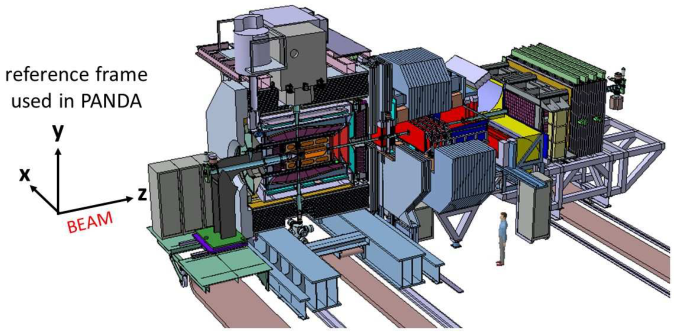


Figure 1. The PANDA full detector

2 The PANDA detector

The PANDA detector is shown in Figure 1. The antiproton beam circulates in the ring along the z-axis in the positive direction. The central part of the detector is inside a 2 Tesla superconducting solenoid. The target system blows frozen hydrogen into the beam line. Inside the solenoid from about 3 cm up to 15 cm away from the beam axis the MicroVertex Detector (MVD) is located, consisting of pixel silicon planes and double-sided strip planes (see Figure 2), the Straw proportional Tubes Tracker (STT) shown in Figure 3, a time of flight tile scintillator detector, a ring imaging silica Čerenkov detector, and a scintillating crystal electromagnetic calorimeter. The muon proportional chambers are inserted between the solenoid iron yoke. The central part is completed by three planes of gas electron multiplier (GEM) detectors covering the very forward angles ($\theta < 11^\circ$), a large backward angle ($\theta > 140^\circ$) scintillating crystal calorimeter and a forward silica disk ring imaging Čerenkov detector. Downstream the solenoid the forward particle section of the experimental setup is placed: a dipole with proportional straw chambers inside, a Time of Flight wall made of plastic scintillators, an aerogel ring imaging Čerenkov, and an electromagnetic and hadronic calorimeter.

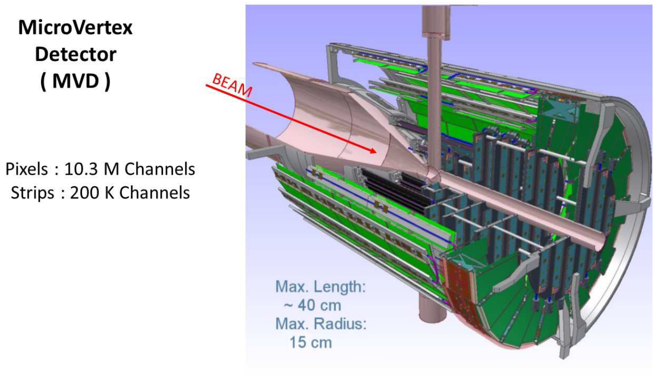
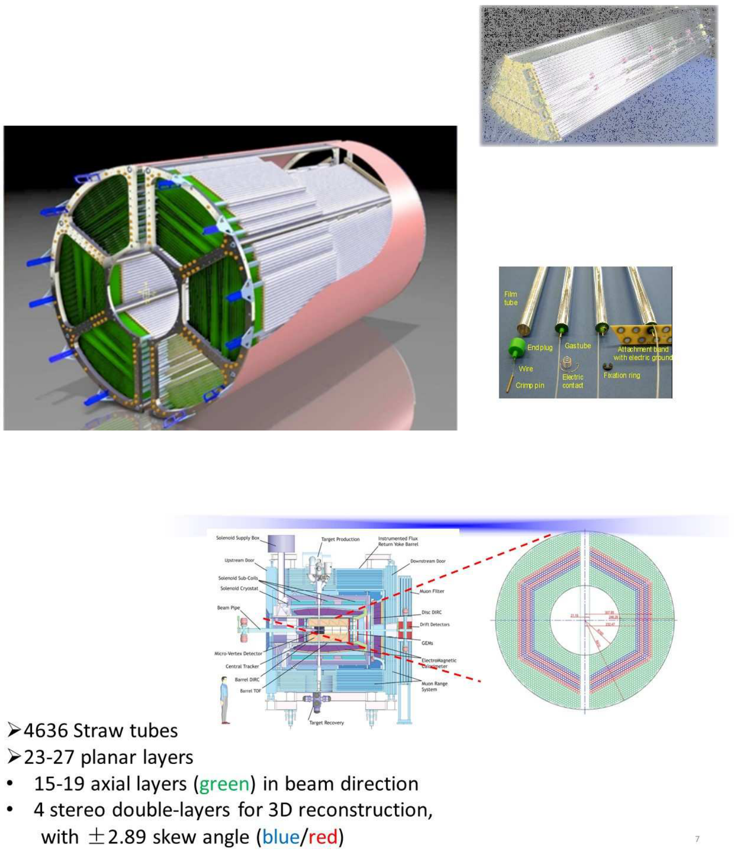


Figure 2. The PANDA MicroVertex Detector



➢4636 Straw tubes
➢23-27 planar layers
- 15-19 axial layers (green) in beam direction
- 4 stereo double-layers for 3D reconstruction, with ±2.89 skew angle (blue/red)

Figure 3. The PANDA Straw Tube Tracker

3 An example of PR running on FPGAs

It is described here an example of a very fast algorithm written for online selection and designed to run on FPGAs. The tracks are searched in the central part of the detector, inside the solenoid, with a road finding algorithm, using only the STT straw hits. First the information on each hit (straw layer and position number, time of arrival of the hit) of the stream of STT hits from the continuous data taking is packed in the FPGA memory in 16 bits per hit (see Figure 4). Then the following algorithms are implemented in the FPGA:

1. The first step consists in finding tracklets with neighbouring STT hits in the x-y-plane combined together (see Figures 3 and 4) starting from the innermost STT axial layers going outwards.

2. The particle trajectory projected on the x-y-plane is a circle; its radius and center position are determined by fitting the tracklets with a circle (χ^2 minimization) in 2 iterations. The $p_\perp$ of the particle at the origin is calculated.

3. The stereo layers hits are associated to the tracklets and the P$_z$ of the particle is found with a χ^2 minimization in two iterations.

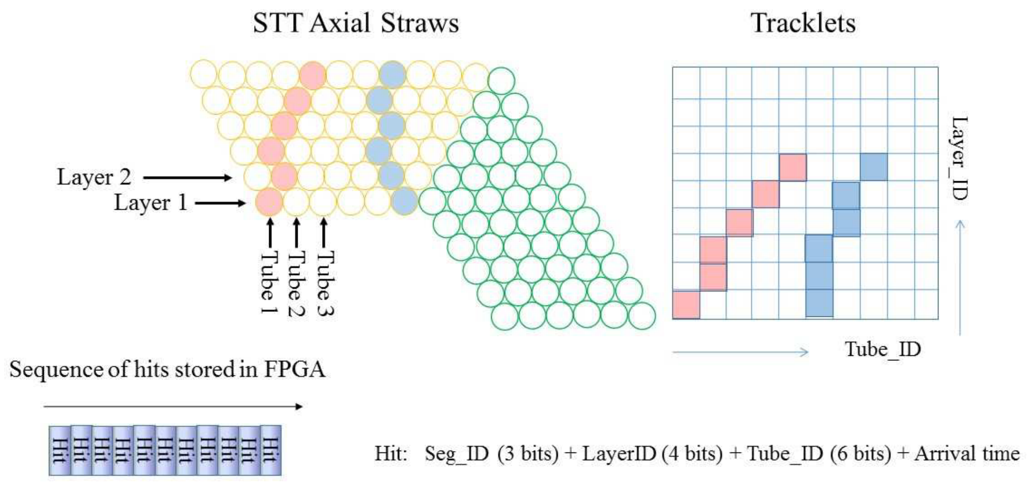

STT Axial Straws
Tracklets
Layer 2
Layer 1
Tube 1
Tube 2
Tube 3
Layer_ID
Tube_ID
Sequence of hits stored in FPGA
Hit
Hit: Seg_ID (3 bits) + LayerID (4 bits) + Tube_ID (6 bits) + Arrival time

Figure 4. Pattern Recognition code running on FPGAs; on the left top part of the STT axial straws is shown with the convention used to identify uniquely each tube; on the top right an example of tracklets found in the STT detector in the Layer$_{ID}$ versus Tube$_{ID}$ plot. At the bottom the detail of the sequentially stored hit information is shown.

Table 1. FPGA clock counts spent in the algorithm for an event with 6 tracks

	Hit Reading	Road Finder	$p_\perp$ extraction	p_z extraction
Clock Counts	100	$50 \times 6 = 300$	$120 \times 6 = 720$	$120 \times 6 = 720$

The algorithm has been implemented in VHDL language on 140 Virtex5 Xilinx FPGAs and it is parallel. The square root operations and the divisions have been sped up using look-up tables.

The algorithm was tested with a stream of STT hits simulated with the standard PANDA simulation package. Events with six muon tracks were generated uniformly in space. An interaction rate of 20 MHz was assumed, thereby stressing the code at the maximum level possible in the PANDA experiment. The high interaction rate produces also pile-up in the recorded hits as a consequence of the STT maximum drift time ($\approx$ 200 nsec), which is longer than the interaction rate. A detail of the clock counts used in each step is given in Table 1 . The total time per 7-track event is 7 μs ; the momentum resolution obtained is $\sigma_{p_\perp} \approx 3.2\%$ for tracks having $p_\perp = 1$ GeV/c, and $\sigma_{p_z} \approx 3.8\%$ for tracks having $p_\perp = p_z = 1$ GeV/c.

This algorithm will be extended to include the MVD hits in the road finding procedure.

4 An example of PR running on GPUs

I describe here an example of parallel code written for GPUs. The tracks are searched in the central parte of the detector, inside the solenoid, using the STT and MVD hits. The strategy used in this algorithm is based on a Hough transform method. After the information of the STT and MVD hits are loaded in the GPU memory the algorithm executes the following steps:

1. in the x-y-plane the x and y coordinates of all MVD hits are transformed to coordinates u and v by a conformal transformation:

$$u \equiv \frac{x^2}{x^2+y^2}; \qquad v \equiv \frac{y^2}{x^2+y^2} \tag{1}$$

consequently the circular trajectory originating from (0,0) in x-y-plane transforms into a straight line going through the MVD points.

2. The same conformal transform is applied to the drift circles of the STT Axial straw hits. Those drift circles transform again to circles in the u-v-plane that are tangent to the particle trajectory (see Figure 5).

3. The Hough plot in the variables θ and R is constructed using the MVD hits and the STT axial straw hits. For each MVD hit a bundle of straight lines (in the u-v-plane) passing through the MVD hit is generated, with the following parametrization:

$$u_i \cos\theta + v_i \sin\theta = R,$$

where u_i and v_i are the MVD hit coordinates. For each STT axial straw hits a similar parametrization is used:

$$u_i \cos\theta + v_i \sin\theta = R \pm d,$$

where u_i and v_i are the coordinates of the center of the straw and d the drift radius of the hit.

An example of the resulting Hough plot for a single-track event is shown in Figure 6. The accumulation point is highlighted in the figure. It is obtained by a peak finding algorithm and allows the determination of the radius and center of the trajectory helix. In the figure the Hough plot consists in 1800×1800 cells in the R, θ variables. This algorithm was parallelized and run on NVIDIA Tesla

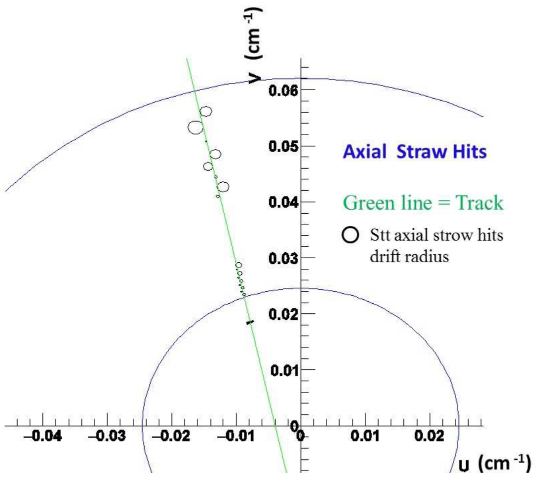


Figure 5. Simulation of a particle trajectory produced at the origin in x-y-plane and conformal-transformed as described in the text

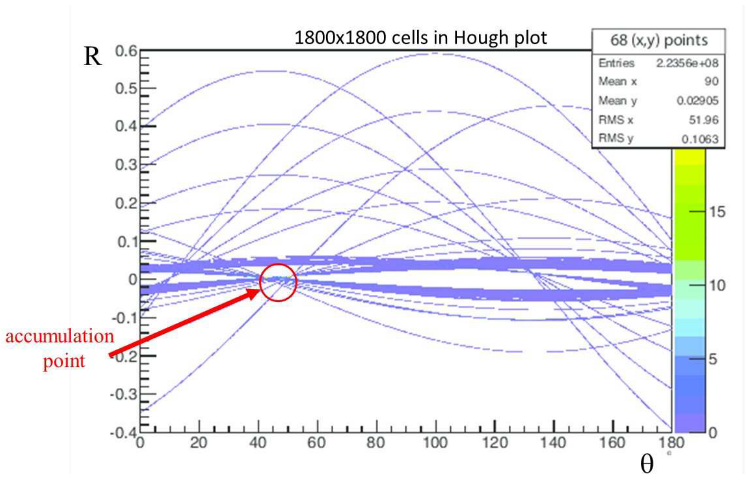


Figure 6. Hough plot obtained with MVD + STT axial straws hits in a 1800 × 1800 Hough plot.

K20K hardware driven by a dedicated CPU. It was written in plain CUDA. The data was fed in the CPU first and then into the memory of the GPU. The performances are relative to the PR algorithm run with the data deposited in the shared memory of the GPUs and are shown in Table 2. So far this algorithm delivers only the $p_\perp$ of the tracks; it will be completed with the inclusion of the calculation of p_z of the tracks, using MVD hits and STT stereo hits.

Table 2. Performance of the parallel algorithm run on NVIDIA Tesla K20K GPUs.

	Hough plot size (N×N) 256 cells	Hough plot size (N×N) 512 cells	Hough plot size (N×N) 1024 cells	Hough plot size (N×N) 2048 cells	Hough plot size (N×N) 4096 cells
% reconstructed tracks	33.0	41.0	55.4	76.0	84.4
% failed tracks	67.0	59.0	44.6	24.0	15.6
% spurious tracks	27.4	14.8	7.2	3.5	0.6
time per event (ms)	0.006	0.013	0.030	0.117	0.507

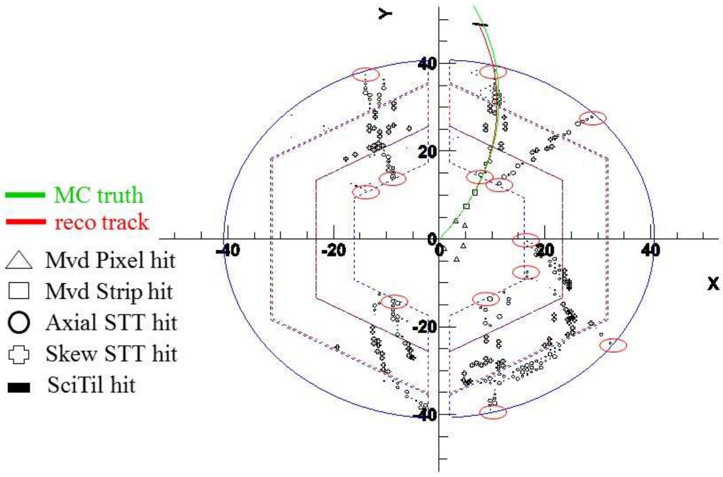


Figure 7. An example of a PANDA single track event projected on the x-y-plane. The green line is the true simulated track, the red line is the reconstructed tracks. Most of the hits in this figure are spurious, remnants of previous (or subsequent) events. Here the (extreme) case of an interaction rate of 20 MHz is show, while at the beginning the PANDA interaction rate is only 2 MHz. The red-circled hits are the STT axial hits from which the tracklets are started.

5 An example of PR running on CPUs

I describe in this section a PR algorithm using the central detector hits to find tracks. It combines the road-finding strategy with the Hough method to maximize efficiency and rejection of spurious hits. Initially it was designed for offline PR and run on CPUs. However, parallelization and implementation on GPUs is foreseen for a possible use online as well.

The first step is finding the tracklets in x-y-plane by using the STT axial straw hits only. They are formed starting from hit at the boundary of the STT detector (red-circled hits in Figure 7) and looking for adjacent hits. In the figure one can notice the large number of spurious hits not belonging to any real track. Those are caused by remnants of tracks from previous events of subsequent events. In PANDA it is one of the major problems encountered in a PR algorithm using the STT detector. It is caused by its large maximum drift time (about 200 ns). Figure 7 is an example of pile-up event in the (extreme) case of interaction rate of 20 MHz, a situation possibly occurring only at a later stage of PANDA. At the start-up of PANDA the interaction rate will be only 2 MHz.

The algorithm proceeds to calculate the conformal transformation in Equation (1), thereby transforming the circular trajectory in the x-y-plane into a straight line in the u-v-plane. A first fast calculation of the radius and center of the trajectory helix is done with a Hough transform plot on the straight trajectory in the u-v-plane. Using this first result the algorithm attempts to add to the tracklet more STT axial straw hits and MVD hits close enough to the trajectory. A χ^2-minimization fit to all STT+MVD hits is performed and the $p_\perp$ and charge of the particle are finally established.

The p_z is determined by associating the STT stereo straw hits to the helix. The stereo angle is only 3° and helps selecting the true hits belonging to the tracks. In fact the spurious straws intersect the trajectory cylinder in unphysical regions (too far back or too far forward). When projected on the trajectory cylinder the trajectory is a straight line (see Figure 8). The MVD hits look like points,

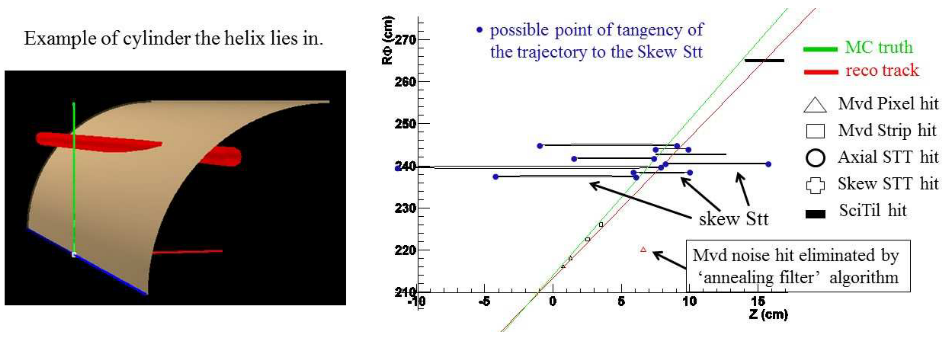


Figure 8. Left: cartoon showing an example of the intersection of a stereo STT straw tube (red cylinder) with the cylinder on which the helix trajectory lies (yellow surface); right: projection on the later surface of the helix cylinder of a helix trajectory. The stereo STT straw (≡ skew STT) intersections look like ellipses squeezed in the vertical direction because of the small stereo angle (3 degrees). One can notice one MVD spurious hit that was eliminated by an annealing filter type of algorithm.

while the stereo STT straw tube intersections look like ellipses squeezed in the vertical direction because of the small stereo angle (3 degrees), with two possible points of intersection with the helix trajectory (blue points in Figure 8). A straight line χ^2-fit delivers the p_z of the track. All possible combinations of stereo STT straw tube intersection points are taken into account in the fit and the solution corresponding to the best χ^2 is chosen. At this point the trajectory is completely determined and a last round of elimination of spurious hits is performed with a simple proximity selection.

Because of the pile-up mentioned earlier, the algorithm may find many spurious tracks (Figure 7 in the worst case of 20 MHz interaction rate). Consequently two cleanup procedures have been written rejecting a track if its hits are not continuous (spatially) in the MVD detector (in the first procedure) or in the MVD and STT detectors (in the second procedure). The performance of this PR is summarized in Tables 3 and 4, calculated with simulated events in which an interaction rate (and consequent pile-up of hits) of 2 MHz was assumed. Events were simulated with 1 to 8 (muon) tracks. All results are summarized here for the two cleanup procedures.

The CPU time spent per track per event on an Intel i7-2600K 64 bits 3.4 GHz core is shown in Table 5 for 2 MHz and 20 MHz interaction rate. The code is not yet parallelized and another factor of 5–10 in speed is expected to be gained when running on a GPU.

6 Other algorithms implemented in PANDA

In PANDA many other Pattern Recognition or tracking algorithms have been written during the years and unfortunately describing them would require too much space. The following is a partial list of them.

- A PR using the MVD hits to find radius and center of the helix cylinder of a track not necessarily produced at the vertex. This algorithm is based on the projection of the circular trajectory on a Riemann sphere (or paraboloid) and can in principle be used to find also secondary vertexes.
- Cellular Automaton types of algorithms that use the MVD and STT hits have been written and implemented on CPUs and GPUs;
- A Hough transform method implemented on GPUs that uses MVD and STT axial straw hits only. This procedure uses an alternative parametrization of the particle circular trajectory in the x-y-plane.

Table 3. Performance of the algorithm run on CPUs assuming an interaction rate of 2 MHz.

MVD hit only cleanup procedure	
True Tracks found	from 92.6% to 98.9 %
STT True Hits found in True Tracks	from 97.8% to 99.7 %
MVD True Hits found in True Tracks	from 98.2% to 99.7 %
Spurious Tracks per event	from 0.26 to 0.33
CPU time per event per track	from 0.26 to 0.33
MVD+STT hit cleanup procedure	
True Tracks found	from 92.2% to 98.5 %
STT True Hits found in True Tracks	from 98.7% to 99.7 %
MVD True Hits found in True Tracks	from 98.2% to 99.7 %
Spurious Tracks per event	from 0.07 to 0.11
CPU time per event per track	from 0.26 to 0.33

Table 4. Performance of the algorithm run on CPUs assuming an interaction rate of 20 MHz.

MVD hit only cleanup procedure	
True Tracks found	from 92.2% to 98.5 %
STT True Hits found in True Tracks	from 95.6% to 98.5 %
MVD True Hits found in True Tracks	from 94.8% to 98.6 %
Spurious Tracks per event	from 2.65 to 2.77
MVD+STT hit cleanup procedure	
True Tracks found	from 66.7% to 77.5 %
STT True Hits found in True Tracks	from 95.7% to 97.2 %
MVD True Hits found in True Tracks	from 95.1% to 97.6 %
Spurious Tracks per event	from 0.58 to 0.89

- An algorithm using only STT hits to find the T_0 ($\equiv$ the time when an event was produced in the primary interaction) of an event; it was implemented on CPUs and also on GPUs.
- Algorithms written in order to find the secondary vertexes;
- PR code for tracks hitting the GEMs (inside the solenoid, in the forward direction).
- PR code for the very forward part of the PANDA detector (trackers in the dipole magnet).

Table 5. CPU time consumption per track per event for an interaction rate of 2 and 20 MHz; MVD+STT hit cleanup procedure.

	2 MHz interaction rate	20 MHz interaction rate
1-track event	1.6 msec	6.5 msec
4-track event	5.0 msec	9.5 msec
8-track event	9.5 msec	13.5 msec

References

[1] PANDA Collaboration (M.F.M. Lutz et al., *Physics Performance Report for PANDA: Strong Interaction Studies with Antiprotons* (2009), arXiv:0903.3905v1 [hep-ex]