A Problem concerning ROOT Performance

Yang Yifan

IHEP

September 4, 2018

The Problem

For the following codes, the ROOT randomly becomes too slow to work with.

```
528             //root (intepreter?) performance problem
529             for (int k=0;k<idxmc;k++){
530                     //psip
531                     if(pdgid[k]==100443) has_100443=1;
532                     if((pdgid[k]==22)&&(pdgid[motheridx[k]]==100443)) has_g1=1;
533
534                     // gKKpi0 signal
535                     if((pdgid[k]==20441)&&(pdgid[motheridx[k]]==100443)) has_20441=1;
536                     if((pdgid[k]==321)&&(pdgid[motheridx[k]]==20441)) has_Kp=1;
537                     if((pdgid[k]==-321)&&(pdgid[motheridx[k]]==20441)) has_Km=1;
538                     if((pdgid[k]==111)&&(pdgid[motheridx[k]]==20441)) has_pi0=1;
539
540                     // gXc1 mode
541                     if((pdgid[k]==20443)&&(pdgid[motheridx[k]]==100443)) has_20443=1;
542                     if((pdgid[k]==321)&&(pdgid[motheridx[k]]==20443)) has_Kp=1;
543                     if((pdgid[k]==-321)&&(pdgid[motheridx[k]]==20443)) has_Km=1;
544                     if((pdgid[k]==111)&&(pdgid[motheridx[k]]==20443)) has_pi0=1;
545
546                     // gXc2 mode
547                     if((pdgid[k]==445)&&(pdgid[motheridx[k]]==100443)) has_445=1;
548                     if((pdgid[k]==321)&&(pdgid[motheridx[k]]==445)) has_Kp=1;
549                     if((pdgid[k]==-321)&&(pdgid[motheridx[k]]==445)) has_Km=1;
550                     if((pdgid[k]==111)&&(pdgid[motheridx[k]]==445)) has_pi0=1;
551
552                     // gXc0 mode
553                     if((pdgid[k]==10441)&&(pdgid[motheridx[k]]==100443)) has_10441=1;
554                     if((pdgid[k]==321)&&(pdgid[motheridx[k]]==10441)) has_Kp=1;
555                     if((pdgid[k]==-321)&&(pdgid[motheridx[k]]==10441)) has_Km=1;
556                     if((pdgid[k]==111)&&(pdgid[motheridx[k]]==10441)) has_pi0=1;
557
558                     //gamma from pi0
559                     if((pdgid[k]==22)&&(pdgid[motheridx[k]]==111)) has_gpi0+=1;
560
561             }
562
563             if (has_100443 && has_20441 && has_g1 && has_Kp && has_Km && has_pi0) event_type=1;//signal
564             if (has_100443 && has_20443 && has_g1 && has_Kp && has_Km && has_pi0) event_type=2;//gXc1
565             if (has_100443 && has_445 && has_g1 && has_Kp && has_Km && has_pi0) event_type=3;//gXc2
566             if (has_100443 && has_10441 && has_g1 && has_Kp && has_Km && has_pi0) event_type=4;//gXc0
567
```

How Slow?

The problem can mean a few hours of machine time for each try. Not practical to work with.

Program Running Time for the first 10,000 events

- code working: around 20 seconds
- code skipped: around 20 seconds
- code commented out: around 0.5 second

Which Part of the Code is Problematic?

Tried locating the problematic code by singling out each line and compare the run time, found that many lines can cause problem alone.

The run time always jump between 1 and 20, never something halfway.

But Similar Code Works Fine.

Curiously, similar code in former part of the same script worked well.

```
// filter out signal events
if (FILTER_SIGNAL==1){
        int has_100443=0;       //psip
        int has_20441=0;        //etac2s
        int has_g1=0;
        int has_pi0=0;
        int has_gpi0=0;
        int has_Kp=0;
        int has_Km=0;
        for (int k=0;k<idxmc;k++){
                if(pdgid[k]==100443) has_100443=1;
                if((pdgid[k]==20441)&&(pdgid[motheridx[k]]==100443)) has_20441=1;
                if((pdgid[k]==22)&&(pdgid[motheridx[k]]==100443)) has_g1=1;
                if((pdgid[k]==321)&&(pdgid[motheridx[k]]==20441)) has_Kp=1;
                if((pdgid[k]==-321)&&(pdgid[motheridx[k]]==20441)) has_Km=1;
                if((pdgid[k]==111)&&(pdgid[motheridx[k]]==20441)) has_pi0=1;
                if((pdgid[k]==22)&&(pdgid[motheridx[k]]==111)) has_gpi0+=1;
        }
        //if (i==1) printf("\n%d,%d,%d,%d,%d,%d\n",has_100443,has_20441,has_g1,has_Kp,has_Km,has_pi0);
        if (!(has_100443 && has_20441 && has_g1 && has_Kp && has_Km && has_pi0)) continue;

}
```

The Same Problem also Happened in My Luminosity.C

The following code runs too slow (> 24h) unless randomly comment out a few lines (around 2h).

```
                    if (IDX_ENERGY>0){
                            c1 = (Ep_emc>(effective_ecms*1.55/4.19)) && (Em_emc>(effective_ecms*1.55/4.19));
                            c2 = (fabs(ctp_mdc)<0.8) && (fabs(ctm_mdc)<0.8);
                            c3 = (pp>(effective_ecms*1.95/4.19)) && (pm>(effective_ecms*1.95/4.19));
                            c4 = (mee>3.8);
                            if (idx_energy==21 || idx_energy==22 || idx_energy==32) c4=(mee>3.69);          // different cut in generator for lower energy points
                    }else{  //method 2, only use EMC info                   // not supported since IDX_ENERGY is no longer effective
                            c1 = (Ep_emc>(effective_ecms*1.8/4.19)) && (Em_emc>(effective_ecms*1.8/4.19));
                            c2 = (fabs(ctp_emc)<0.8) && (fabs(ctm_emc)<0.8);
                            c3 = (fabs(deltaphi)<40) && (fabs(deltaphi)>5);
                            c4 = 1;
                    }

                    // Altered cuts based on ics
                    if (ics==1)     c1 = (Ep_emc>(effective_ecms*1.7/4.19)) && (Em_emc>(effective_ecms*1.7/4.19));  // ics=1: altered E cut
                    if (ics==15)    c1 = (Ep_emc>(effective_ecms*1.8/4.19)) && (Em_emc>(effective_ecms*1.8/4.19));  // ics=1: altered E cut
                    if (ics==2)     c2 = (fabs(ctp_mdc)<0.75) && (fabs(ctm_mdc)<0.75);      // ics=2: altered cos(theta) cut
                    if (ics==16)    c2 = (fabs(ctp_mdc)<0.77) && (fabs(ctm_mdc)<0.77);      // ics=16: altered cos(theta) cut
                    if (ics==3)     c3 = (pp>(effective_ecms*2./4.19)) && (pm>(effective_ecms*2./4.19));    // ics=3: altered P cut
                    if (ics==4){
                                    c4 = (mee>3.85);
                                    if (idx_energy==21 || idx_energy==22 || idx_energy==32) c4=(mee>3.70);

                    }

                    if (ics==11)    c1=1;
                    if (ics==12)    c2=1;
                    if (ics==13)    c3=1;
                    if (ics==20)    c4=1;

                    if (ics==7)     c5=(tofp_t>1e-10 &&tofm_t>1e-10);
                    if (ics==14)    c4=(fabs(deltaphi)<40) && (fabs(deltaphi)>5);
//                  if (ics==8)     c2 = (fabs(ctp_emc)<0.8) && (fabs(ctm_emc)<0.8);

                    if (ics==8||ics==19){           //tang cuts
                            c1 = (Ep_emc>(0.5*0.73*effective_ecms)) && (Em_emc>(0.5*0.73*effective_ecms));
                            c2 = (fabs(ctp_mdc)<0.8) && (fabs(ctm_mdc)<0.8);
                            c3 = (pp>(0.5*0.93*effective_ecms)) && (pm>(0.5*0.93*effective_ecms));
                            if (ics==19)
                            c4 = (fabs(tofp_t)<1e-5 || fabs(tofm_t)<1e-5 || fabs(tofm_t-tofp_t)<4);
```

Conclusion

- ROOT script randomly becomes too slow.
- Educated guess of the cause: some limit of ROOT's C++ interpreter, probably when dealing with a large number of if sentences.
- Can't safely evade the problem by changing the C++ code. Need to switch to python.