Profilování CPU implementace KCF trackeru

Karafiát Vít

16. srpna 2017

Obsah

1 Úvod

Tento dokumnet obsahuje výsldky profilování KCF trackeru (`https://github.com/vojirt/kcf`). K profilování byly použity následující programy:

- Perf
- Hotspot `https://github.com/KDAB/hotspot`
- Intel® VTune™ Amplifier XE for Linux
- Valgrind-massif
- Massif-visualizer

2 Postup

Při profilování byl použit datasety z VOT 2016 pod jménem bag,ball1 a car2.`http://data.votchallenge.net/vot2016/vot2016.zip`
Profilování proběhlo na notebooku ThinkPad x220 s Intel(R) Core(TM) i7-2620M CPU @ 2.70GHz a 8GB RAM.

2.1 Perf+Hotspot

První použítý program na profilování byl linuxový program Perf spolu s programem Hotspot, který umožňuje vizualizaci výsledku z Perfu.

Použitý příkaz: perf record –call-graph dwarf -e r534f2e,r53412e ./kcf_vot

R534f2e a r53412e jsou raw hardware event descriptory (Hardware eventy specifické pro danou architekturu CPU.). Kde r534f2e jsou LLC(Last Level Cache)_REFERENCES a r53421e jsou LLC_MISSES pro Intel Sandy Bridge. K zjistění kódů byl použit libpfm4 `https://sourceforge.net/p/perfmon2/libpfm4/ci/master/tree/`.

2.2 Intel® VTune™ Amplifier XE 2017 Update 4 for Linux

Druhý použítý program byl placený profilovací program od Intelu. VTune™ Amplifier (dále jen Amplifier) umožňuje přístup ke všem performance counterum (Speciální registry na mikroprocesorech sloužící k ukládání hardwerových aktivit v počítačových systémech.) bez potřeby je vyhledávat v manuálu k danému procesoru a automaticky udělá i vizualizaci výsledků. Použitá byla analýza na změření LLC-Hit a LLC-Miss.

3 Popis programu

Program je implementace algorimtu z "High-Speed Tracking with Kernelized Correlation Filters"v C++ `http://arxiv.org/abs/1404.7584`. Víc informací o programu je k dispozici na `https://github.com/vojirt/kcf`
Trax část KCF trackeru nebyla použita.

3.1 Třídy

KCF tracker se skládá ze tříd:

- VOT (vot.hpp)
- KCF_Tracker (kcf.h)
- ComplexMat (complexmat.hpp)
- FHoG (fhog.hpp)
- CNFeat (cnfeat.hpp)

VOT

- Načtení prvního snímku z images.txt a souřádnic a velikosti prvního regionu, který ukazuje sledovaný objekt, z region.txt
- Načítání dalších snímků z images.txt
- Zapisování souřadnich regionů na snímcích do output.txt, včetně prvního snímku

KCF_Tracker

- Výpočty souřadnic regionu na snímcích
- Předávání vypočtených souřádnic regionu skrz BBox_c strukturu třídě VOT.

ComplexMat

- Šablona třídy (Pracuje s datovým typem, který je jí dán.)
- Práce s maticemi pří výpočtech v souřadnic regionu.

FHoG

- Histogram of oriented gradients
- `https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients`

CNFeat

3.2 Průběh

Na začatku programu se v main_vot.cpp vytvoří instance třídy VOT, která zpracuje vstupní soubory images.txt, kde jsou adresy snímků a region.txt, kde jsou souřadnice a velikosti prvního regionu na prvním snímku. Poté uloží první region do output.txt pomocí outputBoundingBox(Def:line 127,vot.hpp) a následné spolu s prvním snímkem je vložen jako parametr do inicializace instance KCF_Tracker pod názvem tracker pomocí metody KCF_Tracker::init(Def:line 6,kcf.cpp).V init se také provedé vypsání velikosti vstupního snímku a o kolik se snímek zmenší, protože na výpočty není potřeba moc velký snímek. Po nainicializování se program přesouvá do while cyklu, ve kterém se opakované volá metoda KCF_Tracker::track (Def:line 135,kcf.cpp, dále jen track) a KCF_Tracker::getBBox(Def:line 123,kcf.cpp, dále jen getBBox) spolu s outputBoundingBox, dokud nedojdou snímky v images.txt.Další snímky načítá funkce getNextImage(Def:line 151,vot.hpp). Funkce getBBox předa pomocí struktury BBox_c souřadnice a velikosti vypočteného regionu metodě outputBoundingBox.
V metodě track probíha hlavní část výpočtů. Podle nastavení proměnné m_use_multithreading v kcf.h se buď použije nebo zakáže multithreadová část v track a použíje se klasická singlethreadová implementace.(Nastavení programu umožňují bool proměnné v kcf.h. Line 32-38) Multithreadová část kódu je implementovaná pomocí funkce std::async.(Funkce nebo kód, který se vyskytuje v async beží asychnronně a může nebo nemusí běžet i ve vedlejším vlákně. Viz `http://en.cppreference.com/w/cpp/thread/async`,dále jen async). V této části kódu se volá metoda KCF_Tracker::get_features(Def:line 283,kcf.cpp, dále jen get_features) . Pokud je m_use_linearkernel nastaven na true, volá se pouze Kcf_Tracker::ifft2(Def:line 457,kcf.cpp, dále jen ifft2) v opačném případě se volá navíc ještě KCF_Tracker::gaussian_correlation(Def:line 541,kcf.cpp, dále jen gaussian_correlation). M_use_linearkernel se dále využívá pří výpočtech v track na řádce 263 až 274.
Ve while cyklu se navíc ještě pomocí cv::getCPUTickCount a cvGetTickFrequency počítá průměrný čas na výpočty regionů v track.
Po skončení celého programu se vypíše průměrný výpočetní čas a počet snímků za sekundu.

4 Výsledky

4.1 Perf

Jak již bylo už zmíněné v sekci Postup 2.1, v Perfu byl měřen počet LLC referencí a LLC missů u všech datasetů. Všechny snímky jsou pořízen z programu Hotspot.

Obrázek 1: LLC-References-bag

Obrázek 2: LLC-References-car2

4.1.1 LLC References

Na snímcích 1, 2 a 3 jsou vidět výsledky z Perfu pro LLC-References. Každý z bloků ukazuje o jakou funkci se jedná a velikost znázorňuje kolik % z celkového počtu zaznamenaných LLC-referencí se objevilo v dané funkci. Pokud ze sebe funkce volá další funkci objeví se jako blok v další úrovni, stejně tak i rekurze. U všech datasetů lze vidět velmi podobné výsledky. U všech datasetů se většina LLC-Referencí vyskytovala ve vedlejších vláknech, která se vytvářejí v async části track. Přesná čísla jsou v této tabulce:

Obrázek 3: LLC-References-ball1

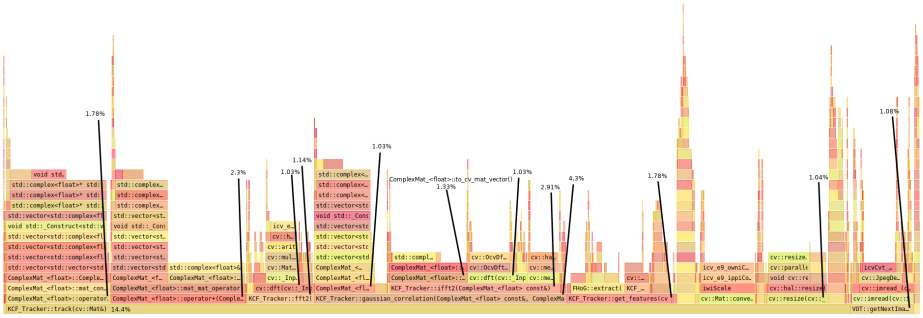
	Vedeljší vlákna(%)	Hlavní vlákno(%)
Bag	73	15.6
Ball1	63.2	19
Car2	71.5	18.3

Tabulka 1: Počet LLC-referencí vedlejsích vláken a hlavního vlákna v datasetech

Do detailů jsou vedlejší vlákna a hlavní vlákno zobrazena na dalších snímcích. Všechny patří pod dataset bag, ostatní datasety nebyly do detailně zobrazeny, protože u všech se vyskytují stejné hotspoty(Místa ve kterých program tráví nejvíc času nebo místa, kde se nejvíce vyskytují sledované veličiny nejvíce,dále jen hotspot). Zvírazněny byly všechny funkce s počtem LLC referencí minimálně 1% a u knihovních funkcí pouze ty,které byly voláné z programu a knihovní funkce volone z kníhovních funkcí už ne.

4.1.1.1 Hlavní vlákno

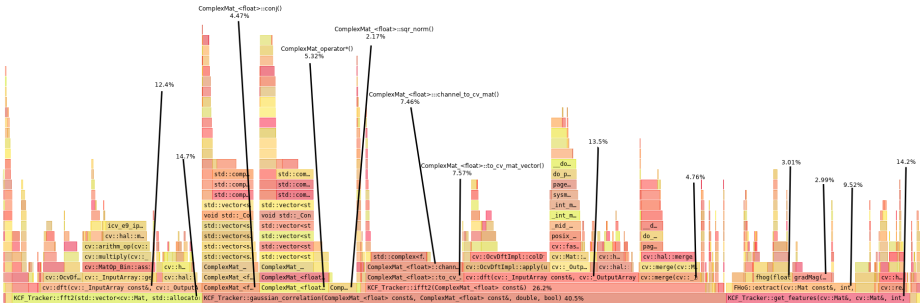
Na obrázku 4 je přiblížení hlavního vlákna pro subset bag. Nejvíce LLC referencí se objevuje ve funkci track. Zde je hlavní hotspot gaussian_correlation,která se používá i mimo async část, pak operátory z třídy ComplexMat, Fourierova transformace ve funkci fft2 a funkce get_features. Většina zbylých LLC referencí je ve funkci getNextImage, která se stará o načítaní snímků. Zde patří z 1.08%, 1.07% LCC referencí knihovní funkci cv::imread.



Obrázek 4: Hlavní vlákno-bag:LLC-Reference

4.1.1.2 Vedlejší vlákna

Stejně jako u hlavního vlákna lze vidět na obrázku 5 se většina LLC referencí u vedlejších vláken vyskytuje ve funkci gaussian_correlation, kde je hotspot funkce ifft2. Další dva hotspoty jsou funkce get_features a fft2. Na obrázku si lze všimnout, že oba výskyty knihovní funkce cv::dft v součtu dávají 25.9% z celkového počtu LLC referencí pro celý program.



Obrázek 5: Vedlejší vlákna-bag:LLC-Rereference

4.1.2 LLC Miss

Obrázek 6: LLC-Miss-bag

Obrázek 7: LLC-Miss-car2

Obrázek 8: LLC-Miss-ball1

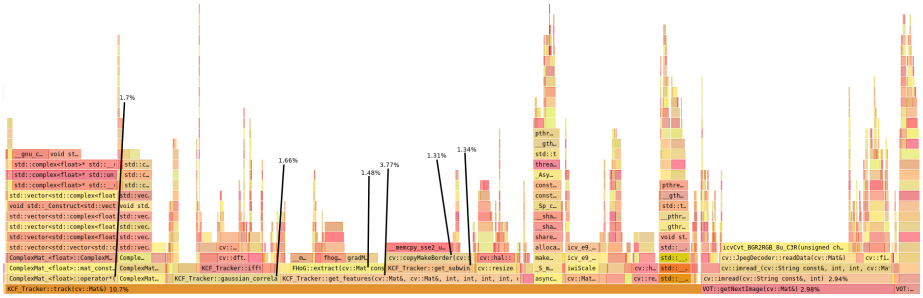
Na snímcích 6, 7 a 8 lze vidět, že výskyt LLC missů se objevuje především ve vedlejších vláknech. Přesná čísla pro všechny datasety jsou v této tabulce:

	Vedeljší vlákna(%)	Hlavní vlákno(%)
Bag	75.2	14.3
Ball1	58.7	19.8
Car2	60.1	25.5

Tabulka 2: Počet LLC-missů vedlejsích vláken a hlavního vlákna v datasetech

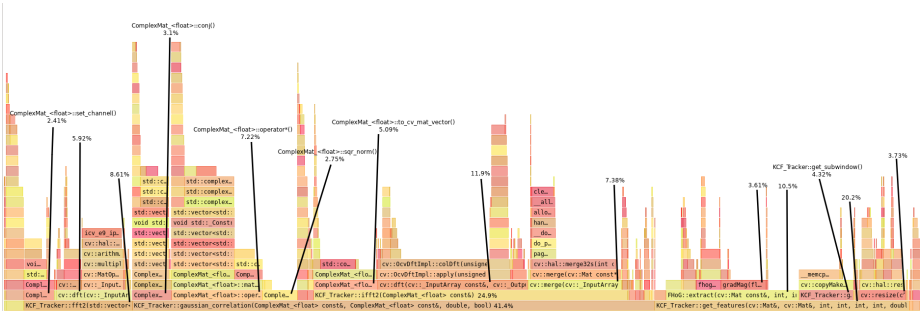
Stejně jako u LLC referencí jsou na dalších snímcích detailně zobrazena vedlejší vlákna a hlavní vlákno pro dataset bag.

4.1.2.1 Hlavní vlákno Přestože nejvíce LLC referencí má v hlavním vlákně ve funkci track funkce gaussian_correaltion, jak bylo vidět na snímku 4, tak na obrázku 9 se nejvíce LLC missů objevuje ve funkci get_features. V ní hlavně u funkcí FHoG::extract(Def:line 20,fhog.hpp,dále jen extract) a KCF_Tracker:: get_subwindow(Def:line 491,kcf.cpp,dále jen get_subwindow). Mimo funkci track se vyskytly LLC missy především při načítání snímků ve funkci getNextImage.



Obrázek 9: Hlavní vlákno-bag:LLC-Miss

4.1.2.2 Vedlejší vlákna U vedlejších vláken se 41.4% všech LLC missů z celého programu vyskytuje ve funkci gaussian_correlation, která je volaná v async části. V ní víc jak polovina všech LLC missů je ve funkci ifft2, kde se provádí Fourierova transformace. Vedle gaussian_correlation je dalším hotspotem znovu funkce get_features, stejně jako v hlavním vlákně. Funkce get_features dohromady s funkcí ifft2 davají v součtu 45.1% všech LLC missů.



Obrázek 10: Vedlejší vlákna-bag:LLC-Miss

4.2 Intel® VTune™ Amplifier XE 2017 Update 4

Analýza proběhla pouze na datasetu bag. Na obrázku 11 je vidět jaký druh Hardware eventů byl měřen i celkový počet zaznamenaných eventů. Amplifier oproti Perfu má mhohem menší počet záznamů, a proto i celkový počet LLC-missů a LLC-Hitů je mnohem menší.

Hardware Events

Hardware Event Type	Hardware Event Count	Hardware Event Sample Count	Events Per Sample
MEM_LOAD_UOPS_MISC_RETIRED.LLC_MISS	12,100,847	121	100007
MEM_LOAD_UOPS_RETIRED.LLC_HIT	29,212,264	584	50021

Obrázek 11: Souhrn analýzy

Na obrázku 12 je detailnější výsledek analýzy seřazen sestupně podle LLC-Missů. Červeným puntíkem jsou označeny funkce z OpenCV a KCF trackeru, ke kterým se mi podařilo najít informace o užití a funkci. Vyjímkou je funkce cv::utils::trace::details::parallelForFinalize (Def:line 962,trace.cpp,dále jen parallelForFinalize). Ke které se mi nepodařilo najít zádne informace v dokumentaci ani fórech OpenCV.

Function / Call Stack	Hardware Event Count by Hardware Event Type		Module	Function (Full)	Source File
	MEM_LOAD_UOPS_MISC_RETIRED.LLC_MISS ▼	MEM_LOAD_UOPS_RETIRED.LLC_HIT			
cv::utils::trace::details::parallelForFinalize	1,500,105	1,350,567	libopencv_core.so.3.3.0	cv::utils::trace::details::parallelForFinalize(cv::utils::trace::details::Region const&)	
cv::OcvDftImpl::colDft	1,200,084	2,250,945	libopencv_core.so.3.3.0	cv::OcvDftImpl::colDft(unsigned char const*, unsigned long, unsigned char*, unsigned long, int, int, bool)	
cv::TLSDataContainer::gatherData	1,200,084	950,399	libopencv_core.so.3.3.0	cv::TLSDataContainer::gatherData(std::vector<void*, std::allocator<void*>>&) const	
gradHist	1,000,070	550,231	kcf_vot	gradHist(float*, float*, float*, int, int, int, int, bool)	gradientMex.cpp
update_blocked_averages	600,042	100,042	vmlinux	update_blocked_averages	
unmap_page_range	400,028	0	vmlinux	unmap_page_range	
icv_e9_ippiMul_32f_C1R	400,028	2,000,840	libopencv_core.so.3.3.0	icv_e9_ippiMul_32f_C1R	
ComplexMat_<float>::mat_mat_operator	400,028	150,063	kcf_vot	ComplexMat_<float>::mat_mat_operator(void (*)(std::complex<float>&, std::complex<float> const&), Compl...	complexmat.hpp
_mm_add_ps	300,021	50,021	kcf_vot	_mm_add_ps(_void, _f3_)	xmmintrin.h
cv::hal::merge32s	300,021	550,231	libopencv_core.so.3.3.0	cv::hal::merge32s(int const**, int*, int, int)	
_ZS10_ConstructIS7complexIfEJRKS1_EEvPT_DpOT0_	300,021	100,042	kcf_vot	_ZS10_ConstructIS7complexIfEJRKS1_EEvPT_DpOT0_	stl_construct.h
cv::MatOp_AddEx::subtract	200,014	0	libopencv_core.so.3.3.0	cv::MatOp_AddEx::subtract(cv::Scalar_<double> const&, cv::MatExpr const&, cv::MatExpr&) const	
free_pcppages_bulk	200,014	100,042	vmlinux	free_pcppages_bulk	
futex_wake	200,014	0	vmlinux	futex_wake	
__libc_malloc	200,014	850,357	libc-2.25.so	__libc_malloc	
cv::resizeAreaFast_Invoker<unsigned char, int, cv::Re	200,014	50,021	libopencv_imgproc.so.3.3.0	cv::resizeAreaFast_Invoker<unsigned char, int, cv::ResizeAreaFastVec<unsigned char, cv::ResizeAreaFastV...	
_mm_mul_ps	200,014	0	kcf_vot	_mm_mul_ps(_void, _f3_)	xmmintrin.h
__mulsc3	200,014	200,084	libgcc_s.so.1	__mulsc3	libgcc2.c
KCF_Tracker::track	200,014	1,200,504	kcf_vot	KCF_Tracker::track(cv::Mat&)	kcf.cpp
_ZNKSt7complexIfE4realB5cxx11Ev	200,014	950,399	kcf_vot	_ZNKSt7complexIfE4realB5cxx11Ev	complex
gradMag	200,014	850,357	kcf_vot	gradMag(float*, float*, float*, int, int, int, bool)	gradientMex.cpp
ComplexMat_<float>::channel_to_cv_mat	200,014	1,200,504	kcf_vot	ComplexMat_<float>::channel_to_cv_mat(int) const	complexmat.hpp
_int_malloc	100,007	300,126	libc-2.25.so	_int_malloc	
strcmp	100,007	150,063	ld-2.25.so	strcmp	
sysmalloc	100,007	50,021	libc-2.25.so	sysmalloc	
down_write	100,007	0	vmlinux	down_write	
copy_user_generic_unrolled	100,007	0	vmlinux	copy_user_generic_unrolled	
__itt_string_handle_create	100,007	2,601,092	libittnotify_collector.so	__itt_string_handle_create	
__memcpy_sse2_unaligned_erms	100,007	350,147	libc-2.25.so	__memcpy_sse2_unaligned_erms	
vmacache_find	100,007	50,021	vmlinux	vmacache_find	
cv::_OutputArray::create	100,007	0	libopencv_core.so.3.3.0	cv::_OutputArray::create(int, int const*, int, int, bool, int) const	
__pthread_disable_asynccancel	100,007	0	libpthread-2.25.so	__pthread_disable_asynccancel	
func@0x13610	100,007	0	libittnotify_collector.so	func@0x13610	
hogChannels	100,007	0	kcf_vot	hogChannels(float*, float const*, float const*, int, int, int, float, int)	gradientMex.cpp
lockref_get_not_dead	100,007	0	vmlinux	lockref_get_not_dead	
__call_rcu.constprop.66	100,007	0	vmlinux	__call_rcu.constprop.66	
cv::String::allocate	100,007	0	libopencv_core.so.3.3.0	cv::String::allocate(unsigned long)	
lru_cache_add_active_or_unevictable	100,007	0	vmlinux	lru_cache_add_active_or_unevictable	
perf_output_begin	100,007	0	vmlinux	perf_output_begin	
perf_event_task_tick	100,007	0	vmlinux	perf_event_task_tick	
_mm_set_ps	100,007	0	kcf_vot	_mm_set_ps(float, float, float, float)	xmmintrin.h
_Z13_mm_storeu_psPfDv4_f	100,007	0	kcf_vot	_Z13_mm_storeu_psPfDv4_f	xmmintrin.h
std::_Function_handler<void (std::complex<float>&)	100,007	100,042	kcf_vot	std::_Function_handler<void (std::complex<float>&), ComplexMat_<float>::conj(void) const::{lambda(std::co...	complexmat.hpp

Obrázek 12: Detailní výsledek analýzy

Nejvíce LLC missů amplifier zaznamenal ve funkci cv::utils::trace::details:: parallelForFinalize (Def:line 962,trace.cpp, dále jen parallelForFinalize). Perf tuto funkci nezaznamenal místo ní zaznamenal funkci cv::parallel_for_(Def:line 372,parallel.cpp, dále jen parallel_for_), která slouží podle dokumentací a fór OpenCV k paralelizaci funkcí v OpenCV. Parallel_for_ se používá ve funkc cv::resize, která se vyskytuje ve funkci get_features. Ampliefier tuto funkci zase naopak nezaznamenal vůbec. Funkce cv::OcvDftImpl::colDft(Def:line 2923,dxt.cpp, dále jen colDft) se využívá v cv::dft, která se používá jak v ifft2, tak i v fft2. Ifft2

se navíc ještě vyskytuje i v gaussian_correlation. Tuto funkci Perf také nezaznamenal. ComplexMat_<float>::mat_mat_operator(Def:line 147,complexmat.hpp), která se využívá v přetěžování operátorů v třídě ComplexMat_. GradHist(Def:line 148, gradientMex.cpp) spolu s hogChannels(Def:line 256,gradientMex.cpp) se využívá při výpočtu HoG(Histogram Of oriented Gradient `https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients`) ve funkci fhog(Def:line 298,gradientMex.cpp), která se náchazí v extract, kde se vyskytuje i funkce gradMag(Def:line 59, gradientMex.cpp), která slouží k výpočtu velikosti a orientace gradientu ve všech místech obrázku. Track, jak už bylo zmíněno v popisu programu, je hlavní funkce programu a probíha v ní hlávní část výpočtů tak obsahuje i async část. ComplexMat_<float>::channel_to_cv_mat(Def:line 184,complexmat.hpp) slouží k získání realné a imaginární složky komplexních výsledků pří diskrétní Fourierově transformaci.

4.3 Porovnání výsledků

Výsledky z Perfu a Amplifieru nebyly úplně stejné. Největším rozdílem byla především funkce parallelForFinalize, která měla nejvíce LLC missů v Amplifieru, a kterou Perf vůbec nezaznamenal při profilování. Navíc online není k této funkci dostatek informací od OpenCV a funkce colDft, kterou také zaznamenal pouze Amplifier. U Amplifieru se navíc stejný počet LLC missů a hitů objevil u několika funkcí. Intel má optimalizované analýzy na své architektury procesorů, což může ovlivňovat výsledky stejně tak i to, že Intel přímo nepodporuje Arch linux, na kterém proběhlo profilovní. S Perfem se však ve funkcích get_features a přetížené operátory v třídě ComplexMat_ výsledky podobají. Obě patří mezi funkce s nejvíce LLC-Missy.