Rok akademicki 2015/2016

Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych

Instytut Informatyki



praca dyplomowa INżynierska

Mikołaj Kamiński

Monitorowanie systemów Microsoft Windows przy wykorzystaniu systemu Icinga

Opiekun pracy

dr inż. Piotr Gawkowski

Ocena:

Podpis Przewodniczącego

Komisji Egzaminu Dyplomowego

Kierunek: Informatyka

Specjalność: Inżynieria Systemów Informatycznych

Data rozpoczęcia studiów: 2012.10.01

Podpis studenta

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu 20\_\_ r

z wynikiem

Ogólny wynik studiów:

Dodatkowe wnioski i uwagi Komisji:

STRESZCZENIE

Streszczenie pracy w języku polskim.

Słowa kluczowe: monitorowanie, Icinga 2, sprawdzenia pasywne, liczniki wydajności

THESIS TITLE IN ENGLISH

Summary in English.

overload

threshold

Keywords: monitoring, Icinga 2, passive checks, performance counters

**Spis treści**

[1 Wstęp 6](#_Toc441432933)

[2 System Icinga 2 9](#_Toc441432934)

[2.1 Architektura 10](#_Toc441432935)

[2.1.1 Rdzeń 10](#_Toc441432936)

[2.1.2 Interfejs graficzny 12](#_Toc441432937)

[2.1.3 Przepływ danych 13](#_Toc441432938)

[2.2 Monitorowanie 14](#_Toc441432939)

[2.2.1 Wykonywanie sprawdzeń 14](#_Toc441432940)

[2.2.2 Monitorowanie zdalne 16](#_Toc441432941)

[2.3 Konfiguracja 19](#_Toc441432942)

[2.4 Dodatki graficzne 21](#_Toc441432943)

[3 Monitorowanie systemu Windows 25](#_Toc441432944)

[3.1 Liczniki wydajności 26](#_Toc441432945)

[3.2 Programy współpracujące z systemem Icinga 2 29](#_Toc441432946)

[3.2.1 NSClient++ 29](#_Toc441432947)

[3.2.2 Icinga 2 - Client 31](#_Toc441432948)

[4 Wtyczka diagnostics-windows 33](#_Toc441432949)

[4.1 Założenia 33](#_Toc441432950)

[4.1.1 Wymagania 33](#_Toc441432951)

[4.1.2 Wybrane rozwiązania 35](#_Toc441432952)

[4.2 Architektura 36](#_Toc441432953)

[4.3 Opis modułów 39](#_Toc441432954)

[4.3.1 PerfData Collector 39](#_Toc441432955)

[4.3.2 Moduł obsługi raportów 41](#_Toc441432956)

[4.3.3 Moduł komunikacji z dodatkiem NSCAv2 42](#_Toc441432957)

[4.3.4 Moduł konsumenta dodatku NSCAv2 44](#_Toc441432958)

[4.3.5 Moduł interfejsu graficznego Icinga Web 2 45](#_Toc441432959)

[5 Testy aplikacji 47](#_Toc441432960)

[6 Podsumowanie 51](#_Toc441432961)

[Dodatek A – instalacja systemu Icinga 2 52](#_Toc441432962)

[Dodatek B – tworzenie modułów do Icinga Web 2 56](#_Toc441432963)

[Bibliografia 57](#_Toc441432964)

# Wstęp

Infrastruktura komputerowa stanowi ważną część współczesnych przedsiębiorstw, urzędów i ośrodków badawczych. Każda awaria lub błąd w działaniu może utrudnić lub nawet uniemożliwić funkcjonowanie tych podmiotów, a w konsekwencji narazić je na straty finansowe. W przypadku wystąpienia usterki, użytkownik systemu informatycznego często też nie potrafi udzielić precyzyjnej informacji o problemie. Każdorazowe, indywidualne szukanie przyczyn błędu jest czasochłonne i powoduje wydłużenie się czasu naprawy. Stąd, koniecznym wydaje się zastosowanie systemu monitorującego stan urządzeń wchodzących w skład infrastruktury IT. Pozwala to na szybszą identyfikację problemów, a także, dzięki ciągłej kontroli, zwiększa szanse uniknięcia awarii.

Powszechnie dostępnych jest wiele narzędzi, z których najpopularniejsze to Nagios, Icinga, Cacti oraz Zabbix. Nagios jest systemem dostępnościowym, czyli zorientowanym na badanie aktualnego stanu systemu. Z tego względu nie przechowuje on danych historycznych oraz nie daje możliwości tworzenia wykresów pokazujących tendencji przeprowadzonych odcztów. Posiada on jednak wiele mechanizmów wspierających administratorów w bieżącym monitorowaniu infrastruktury, takich jak wysyłanie powiadomień czy generowanie raportów. System ten oferuje także szeroki zestaw wtyczek zbierających dane o urządzeniach, które jednocześnie sprawiają, że jego konfiguracja jest stosunkowo trudna, szczególnie dla nowych użytkowników. Icinga jest odgałęzieniem Nagiosa, który eliminuje problem ze skomplikowanymi ustawieniami. Ponadto, w porównaniu z poprzednikiem, w tym systemie położono nacisk na zwiększenie wydajności i poprawę interfejsu graficznego. Podczas gdy pierwsza wersja była mocno związana z Nagiosem, to druga wydana w 2014r. została właściwie napisana od nowa. Cacti, w przeciwieństwie do tych dwóch narzędzi, skupia się na udostępnianiu wykresów na podstawie otrzymanych danych. Posiada on jednak znacznie uboższe możliwości, niepozwalające na informowanie użytkowników o niebezpiecznych zdarzeniach. Istotne także jest, że system ten gromadzi dane w bazach cyklicznych, co nie pozwala na dłuższe przechowywanie danych, gdyż po pewnym czasie są one nadpisywane. Ostanie z wymienionych narzędzi, Zabbix, łączy obie najważniejsze cechy wymienionych wyżej systemów, dając możliwość monitorowania infrastruktury jak i szerokiego analizowania zebranych informacji. Liczne funkcjonalności sprawiają, że również jego konfiguracja, mimo umieszczenia w interfejsie graficznym, może być uciążliwa i skomplikowana. Jako jedyny przechowuje on dane historyczne w relacyjnej bazie danych, dając możliwość usuwania nieistotnych odczytów po ustalonym czasie.

W niniejszej pracy jako narzędzie monitorujące wybrałem system Icinga, który jest już wykorzystywana w Instytucie Informatyki. Jego najnowsza wersja wydana w 2014r. wprowadziła wiele ulepszeń związanych zarówno z wydajnością, jak i wygodą użytkowania, do której należy zaliczyć prostszą konfigurację i nowy interfejs graficzny. Dzięki temu system ten cieszy się coraz większą popularnością, co szczególnie widać po rosnącej społeczności internetowej na forach poświęconych monitorowaniu. Fakt, że narzędzie to jest dostępne od stosunkowo niedługiego czasu, sprawia, że do tej pory powstało niewiele opracować poświęconych jego możliwościom. Jednocześnie opublikowana dokumentacja jest bardzo obszerna, przez co osoby zaczynające pracę z tym narzędziem mogą mieć problemy ze zrozumieniem jego działania. Biorąc to pod uwagę, w tej pracy chciałem uporządkować podstawową wiedzę na temat systemu Icinga 2, która pozwoli w łatwy i szybki sposób rozpocząć pracę z tym narzędziem.

Głównym celem pracy było przeprowadzenie analizy możliwości monitorowania systemu Windows za pomocą systemu Icinga. Oprogramowanie firmy Microsoft jest używane na zdecydowanej większości komputerów osobistych (91%[[1]](#footnote-1)), ale także jest często wybieranym produktem na serwerach internetowych, gdzie jego udział w rynku wynosi 32%[[2]](#footnote-2). Tak duża popularność generuje również duże zapotrzebowanie na narzędzia monitorujące urządzenia z systemem operacyjnym Microsoft Windows. Mimo to przegląd dostępnych rozwiązań współpracujących z systemem Icinga 2 wykazał, że nie wykorzystują one wszystkich możliwości udostępnianych przez system Windows, które mogłyby znacznie ułatwić pracę administratorom infrastruktury informatycznej.

Z tego powodu w ramach pracy została także stworzona aplikacja, której celem jest dostarczanie do systemu Icinga 2 raportów z diagnostyk Monitora wydajności uruchomionych w przypadku nadmiernego obciążenia systemu. Program ten mierzy wykorzystanie zasobów komputera na podstawie liczników wydajności, które są windowsowym mechanizmem dostarczającym szczegółowych informacji o komponentach systemu. Do komunikacji z serwerem monitorującym zostały wykorzystane rozwiązania stworzone przez panów Marcina Kubika i Krzysztofa Opasiaka w pracach [10] i [11].

Rozdział 2 pracy opisuje architekturę i działanie systemu Icinga 2 oraz dostępnych dla niego dodatków graficznych. Zaprezentowane informacje pozwalają na skonfigurowanie narzędzia i rozpoczęcie monitorowania. Rozdział 3 przedstawia przegląd możliwości monitorowania systemu Windows. Zawiera on opis najważniejszych istniejących aplikacji kontrolujących wykorzystanie zasobów i przekazujących uzyskane wyniki do serwera Icinga. Następny rozdział stanowi prezentację wykonanego w ramach pracy projektu. Przedstawiono w nim wymagania stawiane przed stworzoną aplikacją oraz uzasadnienie wybranych rozwiązań. Rozdział ten zawiera także opis architektury i implementacji poszczególnych modułów. W rozdziale 5 zostały przedstawione rezultaty testowania programu oraz wynikające z tego wnioski. Ostatnim elementem pracy są dodatki, w których została przedstawiona „krop po kroku” instalacja systemu Icinga 2 wraz z dodatkami, a także instrukcja pozwalająca na tworzenie własnych modułów do interfejsu graficznego Icinga Web 2.

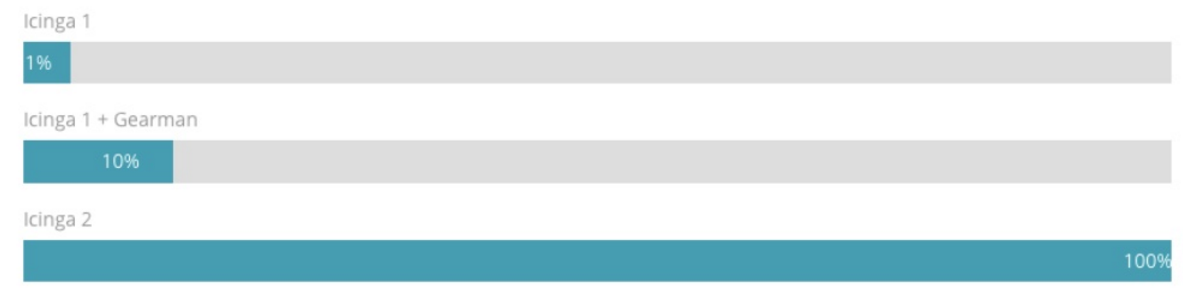
# System Icinga 2

System monitorujący Icinga powstał w 2009 r. jako klon (ang. fork) systemu Nagios. W przeciwieństwie do protoplasty, którego bardziej zaawansowane możliwości były dostępne tylko w wersji płatnej, Icinga jest aplikacją o w pełni otwartym kodzie źródłowym, wydanym na licencji GPL[[3]](#footnote-3) wersji 2. Celem stworzenia nowego oprogramowania było nie tylko poprawienie wad oryginału, ale także dodanie nowych funkcjonalności. Należała do nich większa liczba wspieranych baz danych, a także nowy interfejs graficzny zaprojektowany i napisany w stylu Web 2.0[[4]](#footnote-4). Jednak główna koncepcja monitorowania pozostała zachowana, a twórcy oprogramowania zagwarantowali kompatybilność z wtyczkami przeznaczonymi do Nagiosa. Zapewniło to możliwość monitorowania wielu urządzeń i usług od początku działania systemu. Ponadto Icinga nie jest systemem wyłącznie dostępnościowym, tzn. zorientowanym na ciągłe sprawdzanie stanu infrastruktury informatycznej, ale posiada także cechy systemu analitycznego. Podstawowe zadania stawiane przed systemem Icinga to:

* sprawdzanie dostępności zasobów,
* informowanie użytkowników o awarii,
* zapewnienie obszernych danych analitycznych (ang. business intelligence data).

Dalszy rozwój tego oprogramowania doprowadził do wydania Icingi 2 w 2014 roku. Jednym z głównych założeń było stworzenie systemu skalowalnego oraz pozwalającego na proste rozszerzanie. Wersja 2.0 wprowadziła wiele nowych funkcjonalności oraz zmian architektonicznych, które znacznie zwiększyły wydajność systemu, co przedstawione zostało na rysunku nr 1.

W tym rozdziale zostały zaprezentowane podstawowe informacje o systemie Icinga 2 oraz zmiany, które ten system wprowadził. Pierwsza część poświęcona jest architekturze oprogramowania. Następnie przedstawiono sposoby, w jaki może być monitorowana infrastruktura komputerowa, a także opisano konfigurację, która w nowej wersji została znacząco zmieniona.



**Rys. 2.1** Proporcjonalna liczba sprawdzeń, po których opóźnienie systemu wyniesie 60s. [1]

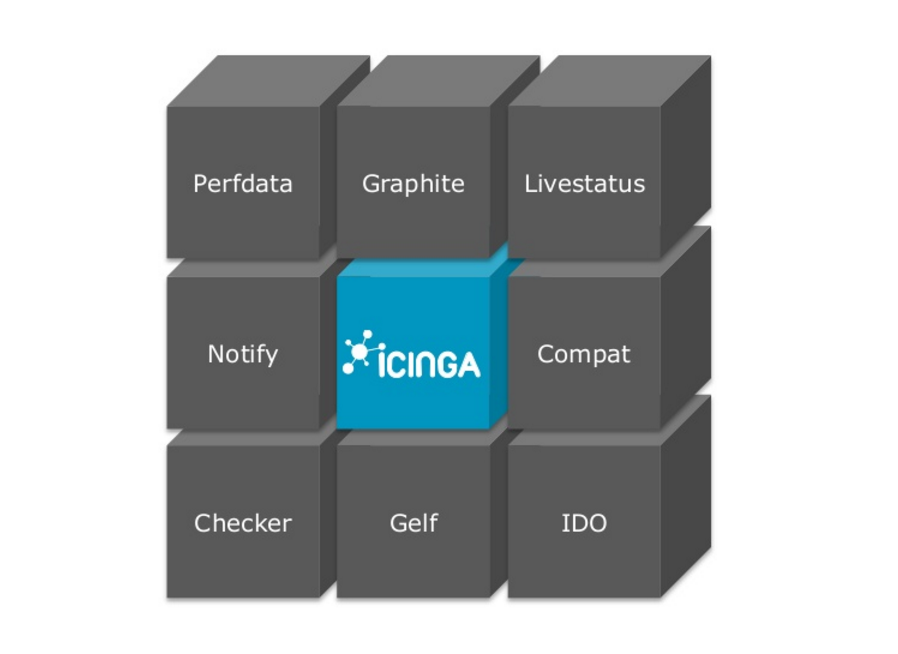
## Architektura

System Icinga 2 składa się z dwóch głównych komponentów: rdzenia monitorującego nazywanego Icinga 2 oraz interfejsu graficznego Icinga Web 2. Osoby odpowiadające za tworzenie tego oprogramowania, wielokrotnie podkreślają [2], że ich celem jest zapewnienie jak największej możliwości integracji z innymi aplikacjami zarówno darmowymi jak i komercyjnymi. Stąd, cechą charakterystyczną obu komponentów jest modułowa budowa pozwalająca na ich proste rozszerzanie.

### Rdzeń

Rdzeń Icinga 2 został napisany w języku C++ z wykorzystaniem zbioru bibliotek Boost. Biblioteki te w znacznym stopniu rozszerzają funkcjonalność języka oraz dają programiście możliwość stosowania bardziej zaawansowanych rozwiązań. Największą wadą tego zestawu jest jednak konieczność zastosowania specjalnego kompilatora oraz długi czas kompilacji programów. Boost pozwala między innymi na tworzenie i zarządzanie wątkami, które stanowią ważny element nowego rdzenia. W poprzedniej wersji monitorowanie odbywało się w jednej pętli programu, co mocno rzutowało na jego wydajność. Od wersji 2 zastosowano wielowątkowość oraz automatyczne rozdysponowanie pracy między wątkami. Dzięki temu, a także innym usprawnieniom, Icinga 2 pozwala na efektywniejszą pracę, co zostało przedstawione na rysunku 2.1.

Rdzeń początkowo był przeznaczony do działania wyłącznie na systemach linuxowych, jednak od listopada 2014 r. , kiedy to została wydana wersja 2.2, aplikacja może być również uruchamiana na systemach operacyjnych Windows oraz Mac OS. Umożliwiło to tworzenie tzw. klastrów monitorujących, które zostały opisane w dalszej części tego rozdziału.



**Rys. 2.2** Moduły wchodzące w skład systemu Icinga 2.

Jak zostało zaznaczone wcześniej, proste rozbudowywanie programu ma zapewniać modułowa budowa. Na rysunku 2.2 zostały przedstawione najważniejsze elementy rdzenia:

* compat – prowadzenie dziennika zdarzeń;
* IDO (Icinga Data Out) – eksport konfiguracji oraz stanu usług do bazy danych, współpracuje z bazami MySQL oraz PostgreSQL;
* gelf – integracja z systemem Graylog, służącym do zbierania i analizowania danych;
* checker – przeprowadzanie sprawdzeń stanu usług;
* notify – wysyłanie powiadomień do użytkowników;
* perfdata – zapisywanie danych wydajnościowych do plików;
* graphite – wysyłanie danych wydajnościowych przez gniazda TCP;
* livestatus – odczytywanie stanu urządzeń i usług z plików „ .dat ”.

Jak wynika z powyższych opisów, część modułów ma podobne zastosowania. To, które z nich będą aktywne, zależy od budowy oraz przeznaczenia systemu i jest ustalane przez użytkownika. Zarządzanie modułami umożliwia CLI[[5]](#footnote-5) dostarczone przez system Icinga.

### Interfejs graficzny

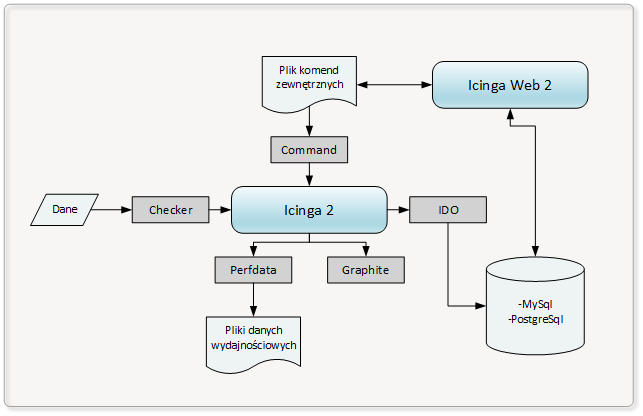
Icinga Web, nazywana interfejsem graficznym, jest w pełni funkcjonalną i niezależną aplikacją internetową (webową). Rozwój technologii internetowych oraz urządzeń mobilnych sprawił, że konieczne się stało również ulepszenie tego elementu systemu Icinga 2. Starsza wersja była mało przyjazna zarówno dla użytkowników jak i programistów. Do szeregu wad należy zaliczyć: nieintuicyjną konfigurację w plikach XML, nadmiarową komunikację oraz skomplikowane API. Utrudniało to także integrację z innymi aplikacjami, co było jednym z głównych założeń nowego systemu.

Icinga Web 2 w części serwerowej została napisana w języku PHP przy użyciu platformy ZendFramework. Użycie takich technik programistycznych jak AJAX, umożliwiło także przesyłanie danych bez odświeżania całej strony, co zmniejszyło ilość transmitowanych danych oraz usprawniło komunikację. Ułatwieniem dla użytkowników jest przeniesienie ustawień do części graficznej oraz udostępnienie kreatora pomagającego przeprowadzić początkową konfigurację. Rozwinięte zostało także zarządzanie użytkownikami. Wprowadzono różne metody autoryzacji oraz nadawanie ról użytkownikom. Umożliwiło to ograniczanie dostępu do monitorowanych urządzeń i serwisów, a także nadawanie uprawnień do korzystania z różnych modułów, wykonywania komend rdzenia Icingi czy modyfikowania konfiguracji. Kolejnym ważnym aspektem z punktu widzenia użytkownika, jest możliwość korzystania z interfejsu także na urządzeniach mobilnych, takich jak smartfony i tablety. Starsza wersja była przeznaczona jedynie dla przeglądarek działających na komputerach lub laptopach. Dla urządzeń przenośnych konieczne było utrzymywania oddzielnej aplikacji mobilnej, z której zrezygnowano w systemie Icinga 2.

Podobnie jak w rdzeniu, współpracę z innymi aplikacjami ma zapewnić modułowa budowa. Twórcom zależało, aby było to maksymalnie proste, więc do dodania własnego komponentu wystarczy, aby spełniał on kilka reguł i był umieszczony w odpowiednim katalogu. Ponadto zostały zostawione dodatkowe mechanizmy pozwalające na jeszcze głębszą integrację niż tylko rozszerzanie o dodatkowe elementy. W wielu miejscach programu zostały umieszczone tzw. zaczepy (ang. hook). Dzięki nim można dodawać własne funkcjonalności lub kod HTML bez ingerencji w kod źródłowy aplikacji. Innym sposobem modyfikowania interfejsu jest możliwość umieszczania własnych łączy internetowych. Konfigurowane może być to dla każdego z monitorowanych urządzeń lub serwisów z poziomu rdzenia (atrybut action\_url).

### Przepływ danych

Na rysunku 2.3 został przedstawiony przepływ danych w podstawowej konfiguracji systemu Icinga 2. Wyniki sprawdzeń trafiające do rdzenia przekazywane są do bazy danych przez moduł IDO, który pozwala na współpracę z bazami MySQL oraz PostgreSQL. Icinga Web 2 prezentowane dane odczytuje bezpośrednio z bazy danych. Komunikacja z rdzeniem wykorzystywana jest wyłącznie do przesyłania komend udostępnianych przez rdzeń Icinga 2. Służy do tego plik komend zewnętrznych, który w rzeczywistości jest potokiem nazwanym. Jest to mechanizm pozwalający na komunikację między procesami w systemach UNIX, co oznacza, że oba komponenty muszą być uruchomione na tym samym urządzeniu. Wśród udostępnianych komend są też takie, które pozwalają na przetwarzanie wyników monitorowania pasywnego, co zostało opisane w następnej części rozdziału. Dodatkowo część danych, tzw. dane wydajnościowe (ang. performance data), trafia do modułów Perfmon lub Graphite. Dane te służą następnie zewnętrznym programom graficznym do generacji wykresów.



**Rys. 2.3** Przepływ danych w podstawowej konfiguracji systemu Icinga 2

## Monitorowanie

Kluczowym zadaniem systemu Icinga 2 jest monitorowanie infrastruktury komputerowej. Odbywa się to poprzesz wykonywanie sprawdzeń stanu usług (ang. service) lub dostępności urządzeń (ang. host) przez wtyczki (ang. plugin), których rezultat jest następnie przetwarzany w rdzeniu. Sprawdzany może być każdy element infrastruktury, który można monitorować, na przykład:

* parametry urządzeń (obciążenie procesora, zajętość pamięci, stan naładowania baterii, itp.),
* dostępność urządzeń (rutery, przełączniki),
* ruch sieciowy,
* usługi sieciowe (HTTP, SMTP, SFTP, itp. ).

### Wykonywanie sprawdzeń

Wtyczki to programy lub skrypty napisane wg określonych reguł, pozwalających na komunikację z systemem Icinga. Do tych reguł należy między innymi format danych zwracanych przez wtyczkę. Wynikiem sprawdzenia zawsze jest stan, określony przez odpowiedni kod liczbowy (tabela 2.1, 2.2). W zależności od tej wartości, system Icinga podejmuje odpowiednie akcje np. wysyła powiadomienie do administratora.

**Tabela 2.1** Stany, w jakich może znajdować się usługa

|  |  |  |
| --- | --- | --- |
| **Stan** | **Kod** | **Opis** |
| OK | 0 | Usługa działa poprawnie |
| Warning | 1 | Usługa przekroczyła wartość ostrzegawczą |
| Critical | 2 | Usługa nie działa lub jest w stanie krytycznym |
| Unknown | 3 | Wtyczka nie mogła określić stanu usługi |

**Tabela 2.2** Stany, w jakich może znajdować się urządzenie

|  |  |  |
| --- | --- | --- |
| **Stan** | **Kod** | **Opis** |
| Up | 0 | Urządzenie jest dostępne |
| Down | 1 | Urządzenie jest niedostępne |

Ponadto, wtyczka może zwrócić dane tekstowe, jednak zajmujące nie więcej niż 8 KB. Dane te mogą także zawierać tzw. dane wydajnościowe. Są one umieszczane po znaku „ | ” i mają zawsze format klucz – wartość liczbowa. Opcjonalnie można także podać jednostkę pomiaru oraz wartości określające progi ostrzegawcze, krytyczne oraz oznaczające maksimum i minimum. Dane wydajnościowe przeznaczone są następnie do generowania wykresów przez zewnętrzne programy.

W kontekście systemu Icinga rozróżniane są dwa typy monitorowania: aktywne i pasywne. Pierwsze z nich, używane najczęściej, polega na cyklicznym odpytywaniu wtyczek o stan usług lub urządzeń przez system Icinga. W drugim przypadku to wtyczka sama inicjuje wykonanie sprawdzenia, po czym w pliku komend zewnętrznych umieszczane jest odpowiednie żądanie przetworzenia otrzymanych wyników. Oprócz wymaganych danych, wymienionych powyżej, wtyczka monitorująca pasywnie musi zwrócić:

* znacznik czasu (w sekundach), określający kiedy dokonano sprawdzenia,
* nazwę wywoływanej komendy,
* nazwę monitorowanej usługi oraz urządzenia, na którym działa.

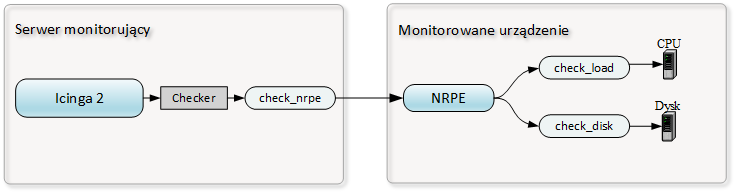
### Monitorowanie zdalne

Warto zauważyć, że w obu wymienionych w poprzedniej części typach monitorowania, wtyczki muszą być uruchomione w kontekście tego samego systemu operacyjnego, co rdzeń Icinga 2. W celu sprawdzania stanu zasobów działających na innych urządzeniach czy też systemach, potrzebne jest użycie mechanizmów pośredniczących między rdzeniem a wtyczką.

Jednym z nich jest protokół SNMP (Simple Network Management Protocol). Służy on do zarządzania infrastrukturą sieciową, a do przesyłania informacji wykorzystuje protokół UDP. W działaniu tego protokołu wyróżniane są dwa typy urządzeń: zarządzające i zarządzane, na których musi być uruchomiony agent SNMP. Zarządca może odczytywać i zmieniać zmienne określające stan danego elementu urządzenia, które zapisane są w bazach MIB[[6]](#footnote-6). Jak wynika z powyższego opisu, mechanizm ten jest przykładem monitorowania aktywnego, jednak protokół SNMP może również służyć do wykonywanie sprawdzeń pasywnych, poprzez wysyłanie komunikatów Trap. Służą one do powiadamiania innych urządzeń, w tym zarządców o wystąpieniu zdarzenia, np. przekroczeniu wartości krytycznej przez jakiś parametr.

Innym rozwiązaniem jest wykorzystanie dodatkowych programów, umożliwiających przesyłanie danych z urządzeń zdalnych. Takie dodatki (ang. addon) zostały stworzone już dla systemu Nagios, ale mogą również działać z systemem Icinga. Ze względu na odmienną charakterystykę przedstawionych typów monitorowania, do każdego z nich wykonano oddzielny dodatek.

Do monitorowanie aktywnego służy NRPE (Nagios Remote Plugin Executor). Program ten składa się z dwóch części: wtyczki działającej na tym samym systemie, co rdzeń Icinga 2 oraz demona uruchomionego na urządzeniu zdalnym. Schemat przeprowadzenia sprawdzenia przy użyciu tego mechanizmu został przedstawiony na rysunku 2.4.



**Rys. 2.4** Schemat działania dodatku NRPE

Moduł sprawdzający (Checker) rdzenia Icinga 2 wywołuje wtyczkę check\_nrpe, której przekazuje informację o usługach, których stan należy sprawdzić. Wtyczka ta łączy się z demonem NRPE na urządzeniu zdalnym (możliwe jest użycie szyfrowanego połączenia SSL), który następnie uruchamia odpowiednie programy sprawdzające. Rezultat ich działania przekazywany jest z powrotem do wtyczki check\_nrpe, a następnie do Icingi.

W przypadku monitorowanie pasywnego używany jest program NSCA (Nagios Service Check Acceptor). Jest to program pełniący rolę serwera, do którego poprzez protokół TCP wysyłane są wyniki sprawdzeń wykonanych na maszynach zdalnych. Następnie NSCA, które działa w kontekście tego samego systemu co Icinga, umieszcza otrzymane rezultaty w pliku komend zewnętrznych. Program ten jednak posiada szereg wad, które wskazał pan Krzysztof Opasiak w pracy „Rozproszone monitorowanie systemów komputerowych”:

* Nieprawidłowy znacznik czasu – dane umieszczane w pliku komend zewnętrznych mają znacznik czasu zmierzony w momencie ich odebrania, a nie wykonania sprawdzenia. Uniemożliwia to przesyłanie danych historycznych, w szczególności dosyłanie danych po wystąpieniu awarii połączenia z serwerem.
* Szyfrowanie symetryczne – ten rodzaj szyfrowania wymusza przechowywanie klucza na każdym monitorowanym urządzeniu, co w znacznym stopniu zwiększa ryzyko jego ujawnienia.
* Brak autoryzacji – program wysyłający dane oprócz klucza szyfrowania, nie musi podawać żadnych danych pozwalających na jego uwierzytelnienie.
* Brak potwierdzeń – NSCA nie wysyła żadnego potwierdzenia otrzymanych danych. Program wysyłający nie dostaje informacji czy rezultaty sprawdzeń dotarły i czy zostały prawidłowo przetworzone.

Z powodu tych wad pan Opasiak w ramach wymienionej wyżej pracy stworzył ulepszoną wersję dodatku NSCAv2, która korzysta również z nowego protokołu przesyłania danych. To rozwiązanie eliminuje opisane niedoskonałości NSCA i wprowadza kilka nowych możliwości. Opracowany protokół jest zbliżony do protokołu SSH i podobnie jak on korzysta z szyfrowania symetrycznego i asymetrycznego. W porównaniu do SSH zostały w nim ograniczone funkcjonalności powodujące przesyłanie nadmiarowej ilości danych (m.in. konieczność posiadania certyfikatów). Dodatkowo, protokół ten nie posiada ograniczenia na długość danych, które są wysyłane przez monitorowaną wtyczkę. Dodatek pana Opasiaka posiada także możliwość dołączenia modułów przetwarzających otrzymane dane. Pozwala to na przekazanie rezultatów wykonania wtyczek do innych miejsc niż tylko plik komend zewnętrznych systemu Icinga.

Ostatnim możliwym mechanizmem monitorowania zdalnych urządzeń, wprowadzonym wraz z systemem Icinga 2 są tzw. klastry. Klaster oznacza kilka lub więcej instancji rdzenia Icinga 2, które ze sobą współpracują (rys. 2.5). Każda z instancji może służyć do innych zadań, dlatego zestaw działających na nich modułów może się różnić. Dodatkowo twórcy systemu Icinga 2 zapewnili możliwość automatycznego przydzielania zadań w wypadku zbyt dużego obciążenia którejkolwiek instancji wchodzącej w skład klastra. Komunikacja między tymi instancjami odbywa się przy pomocy bezpiecznego protokołu SSL. Ponadto użytkownik ma możliwość zdefiniowania w obrębie klastra zon, które pozwalają na wydzielenie jego logicznych części.

Aby wykorzystać możliwości klastra do monitorowania zdalnych maszyn, na każdym monitorowanym urządzeniu trzeba uruchomić rdzeń Icinga 2 zdefiniowany w konfiguracji zony jako satelita. Odpowiada on za lokalne wykonywanie sprawdzeń a następnie ich przesyłanie do głównej instancji w zonie (master). W ramach rdzenia typu master musi działać moduł IDO, który zapisuje spływające do niego odczyty w bazie danych. Elementy w zonie mogą odpowiadać wyłącznie za wykonywanie sprawdzeń (składa się tylko z modułu Checker) lub też być w pełni funkcjonalnymi rdzeniami Icinga 2 (IDO, Notify, …), jednak w obrębie zony wyznaczony jest tylko jeden główny węzeł (master) i tylko on ma możliwość zarządzania bazą danych, konfiguracją, powiadomieniami i wykonywaniem sprawdzeń.

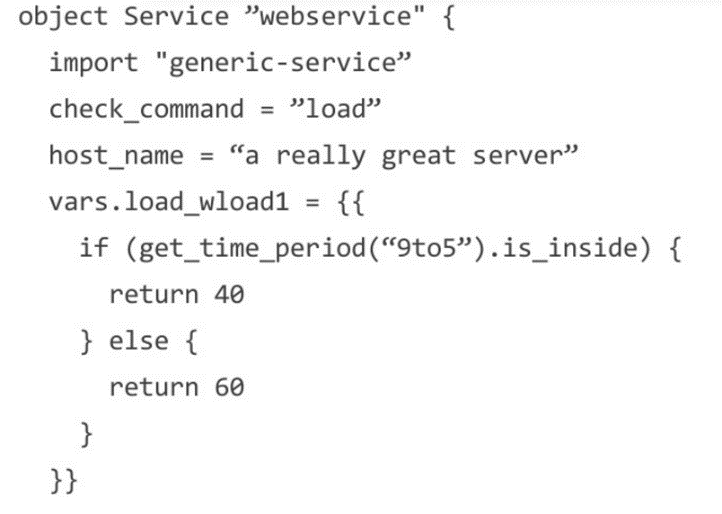
Główną zaletą klastrów jest zapewnienie wysokiej dostępności systemu monitorowania. W przypadku awarii serwera, bazy danych lub któregoś z modułów działających w ramach głównej instancji w zonie, inny rdzeń automatycznie przejmuje jego zadania i staje się węzłem typu master.

## Konfiguracja

**Rys. 2.5** Zona składająca się z węzła głównego i 2 satelitów

Konfiguracja monitorowania w systemie Icinga 2 została całkowicie odmieniona. Mimo to, że dalej mieści się ona w plikach tekstowych, to dostarczono nową i bardzo rozbudowaną składnię opartą na obiektach, przypominającą język skryptowy. Przede wszystkim nowa konfiguracja ma zapewnić użytkownikom wygodną i prostą obsługę. W tym celu autorzy stworzyli dodatki umożliwiające kolorowanie składni w różnych edytorach tekstowych, jak również dostarczyli skrypty pozwalające na automatyczną konwersję plików konfiguracyjnych Icingi 1 oraz Nagiosa do nowego formatu. Przejrzystość nawet dużej i rozbudowanej konfiguracji ma zapewnić podzielenie jej na mniejsze pliki tekstowe. Aby dodać kolejne elementy wystarczy stworzyć nowy plik o dowolnej nazwie i rozszerzeniu „.conf”, a następnie umieścić go w katalogu /icinga2/conf.d, skąd zostanie on wczytany w momencie uruchamiania aplikacji.

Rozbudowana składnia daje użytkownikowi możliwość korzystania z takich mechanizmów programistycznych jak: makra, zmienne, funkcje, instrukcje warunkowe „if .. else” oraz pętle (rys. 2.6). Wszystkie elementy wchodzące w skład systemu Icinga 2 mają odwzorowanie w konfiguracji jako obiekty różnego typu (tabela 2.3)[[7]](#footnote-7). Ustawiając atrybuty obiektów, określa się odpowiednie działanie tych elementów. Ponadto użytkownik może przeprowadzać różne operacje oraz dodawać własne atrybuty, co pozwala na dynamiczną zmianę ustawień w trakcie działania systemu.



**Rys. 2.6** Przykładowa konfiguracja obiektu Service

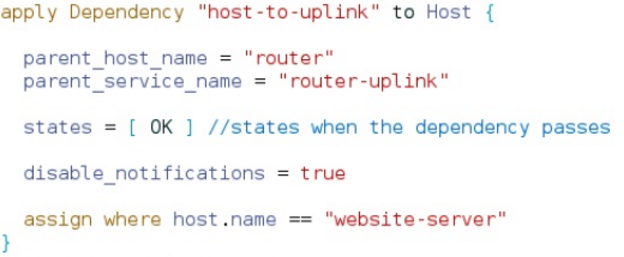
**Tabela 2.3** Przykładowe typy obiektów

|  |  |
| --- | --- |
| **Typ** | **Opis** |
| Dependency | Definiuje zależności między obiektami typu Host i Service |
| Host | Odwzorowuje monitorowane urządzenie |
| IdoMySqlConnection | Definiuje połączenie z bazą danych MySql |
| Notification | Określa w jaki sposób użytkownicy są powiadamiani o awariach |
| PerfdataWriter | Zapisuje dane wydajnościowe, używając określonego makra |
| Service | Odwzorowuje monitorowaną usługę |

Kolejnym rozwiązaniem zwiększającym wygodę administrowania systemem Icinga 2 jest zmniejszenie „objętości” konfiguracji. Użytkownik ma między innymi możliwość tworzenia, a następnie importowania do obiektów szablonów zawierających zestaw generycznych atrybutów (rys. 2.6). Ponadto wprowadzona została możliwość automatycznego przypisywania obiektów (jest to konieczne w przypadku powiązania takich obiektów jak Service i Host czy Notification i Service). Aby to zrobić w definicji obiektu należy użyć słów kluczowych *assign where* i *ignore where* a następnie podać atrybuty, na których podstawie zostanie wykonane dopasowanie (rys. 2.7).

Ważną funkcją systemu monitorującego jest informowanie użytkowników o wystąpieniu sytuacji krytycznej. Sposób, w jaki się to odbywa, określają obiekty typu Notification. Są one przypisane do usług lub urządzeń oraz zawierają informację, w jakich stanach oraz jak należy wysłać powiadomienie. Icinga 2 umożliwia także zdefiniowanie eskalacji powiadomień, czyli sposobu, w jaki będą wysyłane kolejne wiadomości w przypadku braku reakcji na zaistniałe zdarzenie. Aby zapobiec wysyłaniu niepotrzebnych powiadomień użytkownik może ustawić w obiektach Host i Service atrybut *max\_check\_attempts*. Określa on liczbę sprawdzeń, które należy ponownie wykonać w wypadku wystąpienia sytuacji krytycznej, zanim zostanie wysłane powiadomienie. Użytkownik może także definiować zależności między monitorowanymi obiektami (rys.2.7). Jest to przydatne na przykład w sytuacji, w której awarii ulegnie ruter. Mimo to, że wtyczki nie będą w stanie określić stanu usług i urządzeń znajdujących się w sieci „za” ruterem, administrator będzie dostawał powiadomienia jedynie o rzeczywistym problemie, czyli o niesprawności rutera.

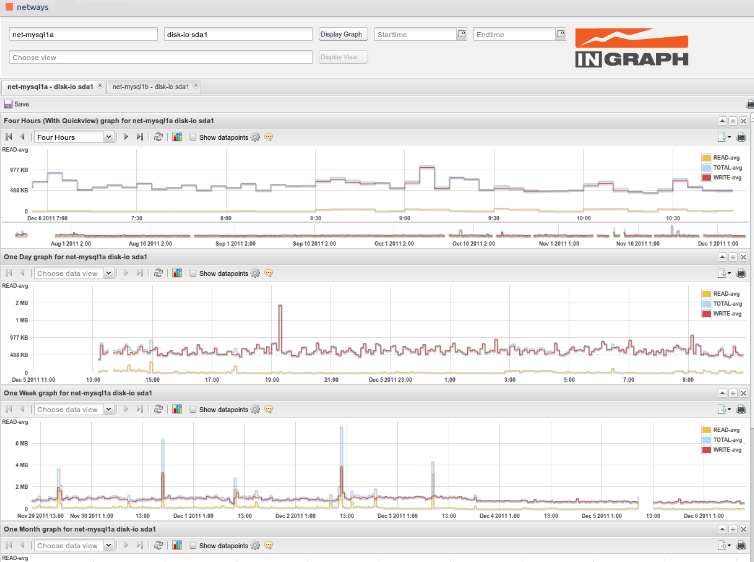
**Rys. 2.7** Konfiguracja zależności



## Dodatki graficzne

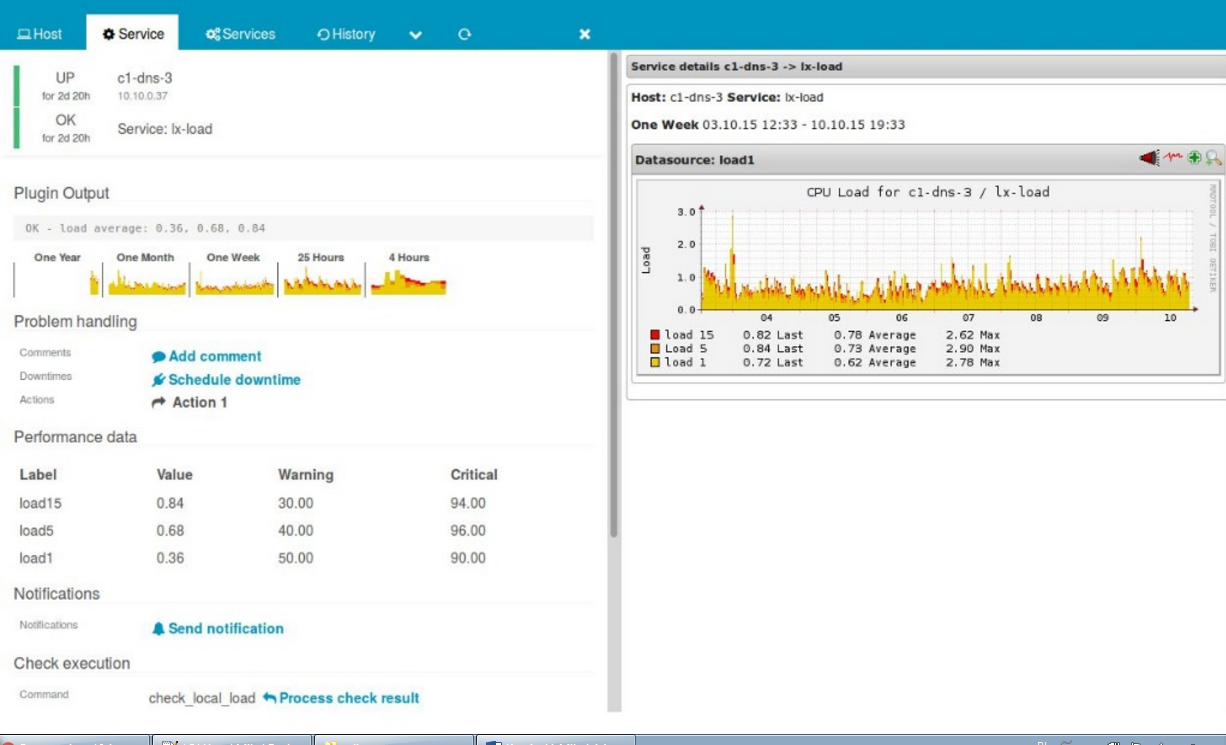
Monitorowanie infrastruktury komputerowej wymaga także analizowania danych historycznych. Obserwowanie zmian wartości parametrów urządzeń w czasie pozwala administratorom przewidzieć możliwe przyczyny awarii i skutecznie im zapobiec. Niestety system Icinga 2 przechowuje w swojej bazie danych jedynie informacje o aktualnym stanie monitorowanych usług. Aby mieć możliwość przechowywania danych historycznych należy użyć zewnętrznych programów. Służą one do rysowania wykresów na podstawie danych wydajnościowych zwracanych przez wtyczki razem ze stanem usług. Dostępnych jest kilka dodatków graficznych, które różnią się sposobem pozyskiwania i przetrzymywania danych, jak również możliwościami edycji generowanych wykresów.

InGraph (rys. 2.8) jest jedynym dodatkiem, który korzysta z relacyjnej bazy danych. Zapobiega to niekontrolowanemu przez użytkownika usunięciu danych, które może mieć miejsce w przypadku użycia baz cyklicznych. Dane potrzebne do rysowania wykresów odczytywane są z plików danych wydajnościowych (generowane przez moduł Perfdata). Wadą programu InGraph jest fakt, iż aktualnie jego rozwijanie zostało zatrzymane, co skutkuje brakiem odpowiedniego modułu pozwalającego na integrację z interfejsem Icinga Web 2.

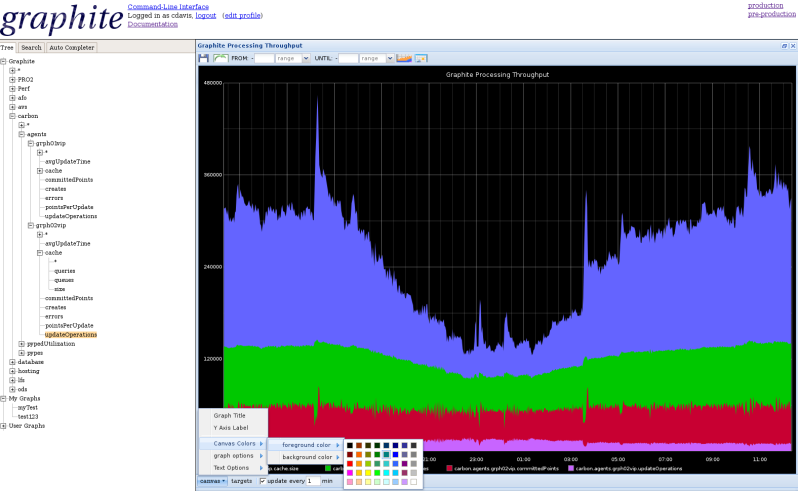


**Rys. 2.8** Interfejs programu InGraph

Nowszymi dodatkami są Pnp4Nagios (rys. 2.9) oraz Graphite (rys. 2.10). Oba te programy korzystają z baz cyklicznych, które aktualnie są najszybszymi bazami NoSQL do przetrzymywania danych szeregów czasowych (ang. time-series data). Pnp4Nagios podobnie jak InGraph korzysta z plików danych wydajnościowych na podstawie których generuje proste wykresy. W momencie pisana pracy jest jedynym dodatkiem integrującym się z interfejsem Icinga Web 2. Graphite jest bardziej zaawansowanym narzędziem. Dane pozyskuje on poprzez protokół TCP (wysyła je moduł GraphiteWriter rdzenia Icinga 2), co umożliwia uruchomienie tego programu na innych serwerach niż te, na których działa Icinga 2. Początko Graphite korzystał z baz RRD (ang. round-robin database), jednak ze względu na ograniczenia przy aktualizacji danych posiadane przez bazy RRD (nie pozwalają one na dosyłanie danych wcześniejszych niż najnowsze; nieprawidłowa obsługa danych dostarczanych nieregularnie) została opracowana i wprowadzona nowa baza danych - Whisper.



**Rys. 2.9** Moduł pnp4nagios w interfejsie Icinga Web 2



**Rys. 2.10** Interfejs programu Graphite

# Monitorowanie systemu Windows

Systemy Microsoft Windows od wielu lat są najczęściej wybieranymi systemami operacyjnymi do komputerów stacjonarnych i laptopów. Ich przewaga nad produktami firmy Apple czy też systemami opartymi na jądrze Linux jest ogromna. W listopadzie 2015r. Microsoft posiadał ponad 91%[[8]](#footnote-8) udziałów w rynku komputerów osobistych, podczas gdy konkurencyjne rozwiązania posiadały odpowiednio 7% i 1,5%. Aktualna sytuacja wskazuje, że ten trend będzie się w najbliższych latach utrzymywał. Po nieprzychylnym przyjęciu przez użytkowników systemu Windows 8, który wymuszał zmianę wielu dotychczasowych przyzwyczajeń, na początku 2015 roku Microsoft wydał nową wersję - Windows 10. System ten powraca do dawnych koncepcji użytkowania i już pod koniec 2015 roku liczba urządzeń, na których jest zainstalowany zbliżyła się do tych z Windows 8. Ponadto brak nowych konkurentów oraz współpraca z wieloma producentami sprzętu, którzy swoje produkty sprzedają z zainstalowanym systemem Windows, pozwala przewidywać, że w najbliższych latach produkt firmy Microsoft będzie nadal liderem wśród systemów operacyjnych.

Ogromna popularność produktu Microsoft’u skutkuje także dużym popytem na zewnętrzne aplikacje działających na ich systemie. Do ich tworzenia został udostępniony interfejs programistyczny Windows API (wcześniej nazywany Win32 API). Dzięki niemu można rozwijać aplikacje w pełni korzystające z właściwości i możliwości kolejnych wersji systemu Windows. Zestaw funkcji dostarczonych przez ten interfejs pozwala na dostęp do innych programów, parametrów komputera, usług sieciowych czy też tworzenia aplikacji okienkowych przy użyciu elementów interfejsu graficznego systemu Windows. W kontekście monitorowania najważniejszą częścią API są funkcje służące do diagnostyki. Dają one programiście możliwość korzystania z liczników wydajności (ang. performance counters). W punkcie 3.1 przedstawiłem sposób ich działania oraz uzyskane dzięki nim możliwości monitorowania.

Systemy operacyjne Microsoft Windows znalazły zastosowanie zarówno w użytkowaniu domowym, naukowym i biznesowym. Szczególnie wykorzystanie go w rozwiązaniach komercyjnych, gdzie każda awaria lub brak dostępności usługi powoduje straty finansowe, spowodowało, że powstało wiele narzędzi monitorujących stan systemu jak i urządzenia, na którym działa. W drugim punkcie tego rozdziału zostały opisane te, które umożliwiają współpracę z systemem Icinga 2.

## Liczniki wydajności

Celem liczników jest informowanie o wydajności systemu, aplikacji, usług i sterowników. Dzięki zebranym przez liczniki danym można zdiagnozować „wąskie gardła” systemu oraz wyregulować działanie programów, aby ich wydajność była jak największa. Sprawdzanie stanu systemu lub urządzeń nie jest jednym możliwym zastosowaniem liczników. Różne aplikacje mogą także z nich korzystać w celu sprawdzenia, jak dużo zasobów jest dostępnych. Na przykład aplikacja korzystająca z karty sieciowej może pobrać z liczników informację, jaką część pasma może wysyłać nie przeszkadzając innym aplikacjom.

Listę wszystkich dostępnych w danym systemie liczników można sprawdzić poprzez wiersz poleceń używając komendy *typeperf* z odpowiednimi argumentami. Pozwala ona również na cykliczne sprawdzanie wartości liczników w określonych odstępach czasu oraz na eksportowanie otrzymanych wartości do plików tekstowych, binarnych lub baz danych.

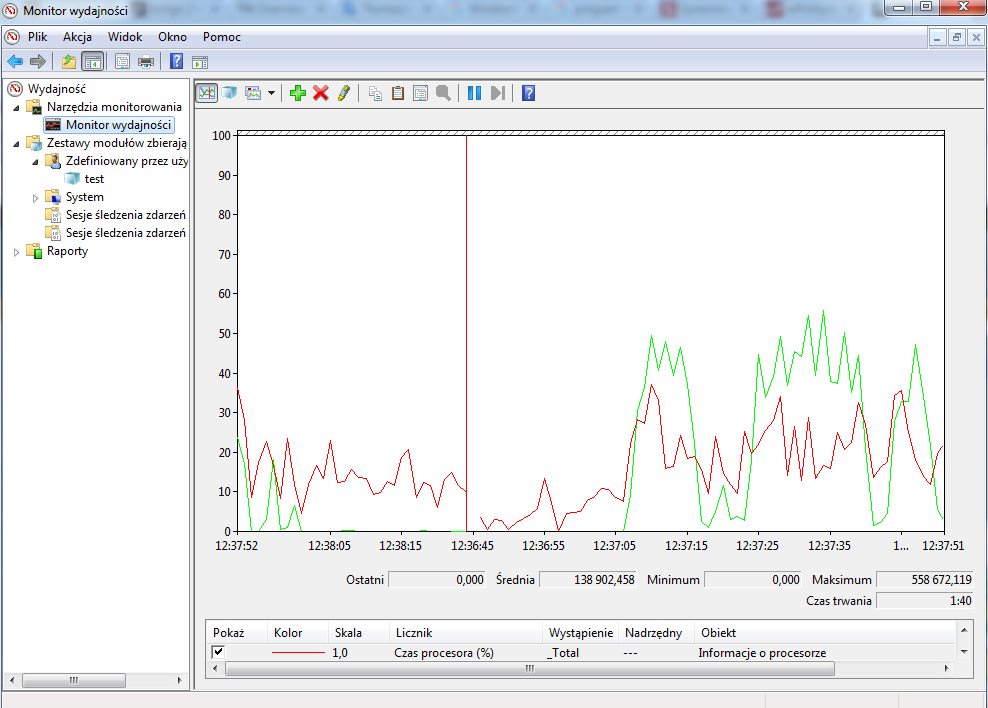
Ze względu na dużą liczbę dostępnych liczników (ponad 1000, dokładna liczba zależy od wersji systemu, komponentów komputera i ilości działających procesów) zostały one podzielone na kategorie odpowiadające usługom i urządzeniom, o których stanie informują (np. procesor, dysk, IPv4, interfejs sieciowy). Dodatkowo w obrębie jednej kategorii mogę być wyszczególnione instancje, które oznaczają logiczne lub fizyczne części monitorowanego zasobu. W przypadku dysku są to partycje, na które jest on podzielony, a w przypadku procesora dostępne rdzenie.

Do najważniejszych elementów, które mogą powodować spowolnienie lub nawet awarię systemu zaliczamy: dysk twardy, pamięć podręczną, działające procesy, procesor oraz usługi sieciowe. W tabeli 3.1 przedstawione zostały najistotniejsze liczniki pozwalające monitorować te krytyczne zasoby. Przed znakiem „\” znajduje się kategoria, a po nim nazwa licznika. Znak (\*) oznacza, że dla danej kategorii należy podać również nazwę instancji, dla której chcemy wykonać pomiary.

**Tabela 3.1** Wybrane liczniki wydajności

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Opis** | **Wartość** |
| Dysk logiczny(\*)\ Wolne miejsce (%) | Sprawdza ile procent wybranego dysku logicznego jest wolne. | > 15% |
| Dysk fizyczny(\*)\ Czas bezczynności(%) | Mierzy jak długo dysk był bezczynny w trakcie próbkowania. | > 60% |
| Dysk fizyczny(\*)\ Średnia długość kolejki dysku | Liczba operacji wejścia/wyjścia oczekujących na wykonanie. Przyczyną zbyt dużej wartości jest zazwyczaj samo urządzenie. | < 2 |
| Pamięć\Dostępna pamięć (MB) | Ilość pamięci dostępna dla wykonujących się procesów. Zbyt mała wartość powoduje zwiększenie stronicowania | > 10% |
| Pamięć\Strony/s | Liczba stron zapisywanych lub odczytywanych z dysku. Nadmierna ilość może powodować wycieki pamięci. | < 1000 |
| Interfejs sieciowy(\*)\ Całkowita liczba bajtów/s | Oznacza ile bajtów jest przekazywanych przez poszczególne karty sieciowe. Wartość ta jest porównywana z dostępną prędkością transmisji karty. | < 70% |
| Interfejs sieciowy(\*)\ Długość kolejki wyjściowej | Ilość pakietów oczekujących na wysłanie. | < 2 |
| Procesor(\*)\Czas procesora (%) | Czas procesora poświęcony na inne wątki niż wątek bezczynności. | < 85% |
| Procesor(\*)\Czas przerwań (%) | Ilość czasu w jakim procesor obsługuje przerwania sprzętowe. Przekroczenie wartości krytycznej oznacza problemy ze sprzętem. | < 15% |
| Pamięć\Bajty w puli niestronicowanej | Wielkość obszaru przeznaczonego na obiekty, które nie mogą zostać wymiecione na dysk. Krytyczna wartość to stosunek do wielkości całej pamięci. | < 75% |

Od systemów Windows Vista oraz Windows Server 2008 użytkownicy mogą korzystać także z „Monitora wydajności” (rys 3.1). Jest to program z interfejsem graficznym, pozwalający na sprawdzanie wartości liczników. Użytkownik może wybrać zestaw liczników, a następnie, na generowanych w czasie rzeczywistym wykresach, obserwować zmiany ich wartości.

Monitor Wydajności daje także możliwość przeprowadzenia diagnostyki, której efektem jest wygenerowanie raportu mówiącego o stanie systemu. Aby uruchomić diagnostykę trzeba najpierw zdefiniować zestaw modułów zbierających dane. Zawiera on informację, jakie komponenty powinne być monitorowane. Domyślnie dostępne są dwa systemowe zestawy, ale zapewniona jest możliwość tworzenia własnych, spełniających specyficzne wymagania użytkownika. Ponadto w ramach konfiguracji takiego zestawu, można określić czas trwania, lokalizację oraz nazwę docelowego katalogu, który zawiera wszystkie pliki generowane w ramach przeprowadzonej diagnostyki. Raport dostępny jest w 2 formatach HTML oraz XML. Zapewnia to możliwość zarówno wygodnego prezentowania danych w przeglądarce, jak i dostępu do poszczególnych pomiarów. Dostępne są także pliki zawierające informację o wartościach liczników wydajności w trakcie trwania diagnostyki. Umożliwiają one wyświetlenie tych danych w formie wykresów w Monitorze Wydajności (Performance Counter.blg) lub w Windows Performance Analyzer (NtKernel.etl).

**Rys. 3.1** Ekran programu Monitor Wydajności

Raport programu Monitor Wydajności jest cennym źródłem informacji o systemie nie tylko z powodu zbierania informacji o stanie liczników, gdyż je można sprawdzić ręcznie na kilka sposobów. Gromadzi on także dane śledzenia zdarzeń oraz informacje o konfiguracji systemu. Dzięki temu możemy się dowiedzieć, jakie aplikacje były uruchamiane oraz czy któreś z nich uległy awarii. Ponadto ważną informacją dostarczaną przez raport jest wskazanie plików oraz procesów odpowiedzialnych za największe wykorzystanie zasobów.

## Programy współpracujące z systemem Icinga 2

Wtyczki są podstawowym sposobem zbierania danych dostępnym w systemie Icinga 2. Mechanizm ten został odziedziczony po systemie Nagios, a Icinga zapewnia kompatybilność z wtyczkami stworzonymi dla poprzednika. Na stronie z oficjalnymi wtyczkami dla obu systemów znajduje się ponad 100 monitorujących systemy Windows. Poziom ich zaawansowania jest bardzo zróżnicowany, od programów sprawdzających pojedyncze parametry do narzędzi o rozbudowanej architekturze i wielu monitorowanych usługach. Dodatkowo została wydzielona sekcja wtyczek korzystających z WMI (Windows Management Instrumentation), czyli stworzonej przez firmę Microsoft implementacji Web Based Enterprise Management Initiative. Niestety strona ta nie zawiera informacji o liczbie pobrań poszczególnych wtyczek, dlatego nie można stwierdzić, które z nich są najbardziej popularne. Ponadto daty udostępnienia kolejnych programów oraz aktywności użytkowników (dodane komentarze) wskazują, że w ostatnich latach wspieranie i promowanie pojedynczych wtyczek zostało zaniechane.

### NSClient++

W momencie pisania pracy, dokumentacje Icinga 2 i Nagios, w celu monitorowania systemów Windows, odsyłały do korzystania z programu NSClient++. Był on też najczęściej polecany w artykułach i blogach dotyczących monitorowania. NSClient++ zaliczany jest do normalnych wtyczek (umieszczony jest na oficjalnej liście), jednak oferowane przez niego możliwości, popularność oraz fakt, że jest on cały czas rozwijany sprawiają, że warto go szczegółowiej omówić.

Celem aplikacji jest zapewnienie wszechstronnego narzędzia do gromadzenia informacji o urządzeniu i przesyłaniu ich do systemu monitorującego. Mimo, że pierwotnie program ten miał być klientem Nagiosa, współpracuje on także z systemem Icinga 2. Ponieważ NSClient++ działa na maszynach zdalnych, aby umożliwić przesyłanie danych do tych systemów trzeba użyć programów pośredniczących. Program może komunikować się z oboma dodatkami do Icingi 2: NSCA oraz NRPE, co umożliwia wykonywanie sprawdzeń aktywnych i pasywnych. Ponadto dokumentacja informuje o innych wspieranych protokołach, dzięki którym można go powiązać nie tylko z systemami Icinga oraz Nagios. Należą do nich:

* NRDP,
* check\_mk,
* Syslog,
* Grphite,
* SMTP,
* CollectD,
* REST.

NSClient++ dostarcza zestaw modułów służących do monitorowania. Składają się na nie skrypty napisane w języku Lua i Python oraz biblioteki DLL. Pozwalają one na sprawdzenia wielu parametrów komputerów, w szczególności tych działających na systemach Windows, gdyż większość komponentów przeznaczona jest właśnie dla tego systemu. Z tego powodu program może także używać z plików DLL, które pozwalają wykorzystać szeroki pakiet funkcji udostępnianych przez natywne biblioteki środowiska Microsoft Windows. Jedną z nich jest biblioteka pdh.dll, która zawiera implementacje funkcji wykonujących operacje na licznikach wydajności. Dzięki temu program NSClient++ ma dostęp do wszystkich najważniejszych parametrów systemu operacyjnego.

Minusem NSClient++ jest jego trudne użytkowanie. Nie posiada on interfejsu graficznego, dlatego wszystkie ustawienia znajdują się w plikach tekstowych. Rozpoczęcie pracy programu i zdefiniowanie podstawowych monitorowanych parametrów nie jest trudne i nie wymaga dużo czasu. W wypadku problemów można także uruchomić tryb debugowania, który w plikach logów umieści więcej informacji o zaistniałych błędach. Mimo to, użyty format *INI* sprawia, że wraz ze wzrostem ilości sprawdzanych parametrów i wykorzystywanych modułów, konfiguracja staje się bardzo nieczytelna i uciążliwa w utrzymaniu. Braki w dokumentacji powodują również trudności w dodawaniu własnych modułów oraz integracji z innymi aplikacjami poprzez wymienione wyżej protokoły. Program jest jednak cały czas rozwijany (stabilna wersja 0.4.3 w momencie pisania pracy), dlatego można się spodziewać, że wymienione wady zostaną wyeliminowane w przyszłych wersjach.

### Icinga 2 - Client

Ostatnim sposobem monitorowania systemu Windows jest zastosowanie klastrów, które zostały omówione w podpunkcie 2.2.2 Monitorowanie zdalne. W przeciwieństwie do programów omówionych powyżej, jest to rozwiązanie stworzone wyłącznie do pracy z systemem Icinga 2. Polega ono na utworzeniu zon, składających się z wielu rdzeni monitorujących. Podstawowymi celami takich pojedynczych instancji systemu Icinga 2 jest możliwość rozłożenia nadmiaru zadań pomiędzy poszczególne elementy oraz zapewnienie wysokiej dostępności w przypadku awarii. Jednak mogą one również pełnić rolę agentów monitorujących urządzenia zdalne. Taka możliwość dla produktów firmy Microsoft istnieje od listopada 2014r. , kiedy to opublikowano wersję 2.2.0 systemu Icinga, przeznaczoną również dla systemów Windows.

Mimo, że w oficjalnej dokumentacji nie znajduje się wiele informacji o wersji działającej na systemach Windows, to jej Twórcy zadbali o wygodę użytkownika poprzez dodanie kreatora umożliwiającego prostą instalację. Odpowiada on także za ustawienie parametrów protokołu SSL oraz automatyczne podpisanie certyfikatów bezpieczeństwa. Po instalacji użytkownik może od razu rozpocząć monitorowanie dzięki dostarczonym w ramach rdzenia wtyczkom. Sprawdzają one informacje na temat:

* zasobów urządzenia (dysku, interfejsu sieciowego, …),
* uruchomionych procesów i usług,
* aktualizacji systemu,
* użytkowników,
* wartości liczników wydajności.

Działanie oraz konfiguracja Icingi na systemie Windows jest identyczna jak w przypadku wersji linuxowych, jednak nie posiada ona wszystkich modułów. Między innymi brakuje modułu Command, odpowiadającego za obsługę pliku komend zewnętrznych, który jest mechanizmem unixowym (potok nazwany). Zatem na systemie Windows nie jest możliwe wykonywanie sprawdzeń pasywnych oraz uruchomienie interfejsu Icinga Web 2, dla których wymagana jest obsługa komend zewnętrznych. W kontekście architektury klastrów, taka instancja rdzenia musi posiadać węzeł nadrzędny, do którego będzie przekazywała otrzymane rezultaty sprawdzeń.

# Wtyczka diagnostics-windows

Systemy Icinga oraz Nagios są dostępne do użytkowania od kilkunastu lat. W tym czasie powstało wiele programów nazywanych wtyczkami, które umożliwiają tym systemom monitorowanie infrastruktury komputerowej. Mimo, że liczba oraz spektrum zastosowań tych aplikacji jest dosyć duże to można zauważyć, że skupiają się one na dostarczaniu informacji o stanie infrastruktury. Widać to w przypadku opisanych w poprzednim rozdziale wtyczek pracujących na systemie Windows, które sprawdzały zużycie pamięci, interfejsu sieciowego, itp. Informacje te pozwalają na zaobserwowanie sytuacji krytycznych czy też przewidzenie możliwych usterek urządzeń. Zapewnia to spełnienie wymagań stawianych przed systemami monitorującymi, jednak nie dostarcza wiedzy o rzeczywistej przyczynie awarii. Administrator musi sam dochodzić, co było powodem niesprawności nadzorowanego systemu. Może to oznaczać bezpośrednie zbadanie urządzenia lub pozyskanie informacji od użytkowników, którzy ze względu na brak fachowej wiedzy, często nie są w stanie dokładnie sprecyzować, jakie zdarzenie wystąpiło. Jest to tym bardziej trudne, jeżeli awaria wystąpiła w przeszłości i w momencie podjęcia działania przez administratora system powrócił do prawidłowego działania.

Przeprowadzona analiza wskazała, że system Windows może dostarczyć informację o przyczynie nadmiernego obciążenia komputera, jednak istniejące wtyczki systemu Icinga tego nie wykorzystują. Z tego powodu w ramach niniejszej pracy stworzyłem aplikację, która pozwoli wypełnić lukę w zastosowaniach dotychczas istniejących wtyczek, tzn. umożliwi przesyłanie raportów z diagnostyk systemu wykonanych w Monitorze Wydajności do Icingi. Przegląd raportu zawierającego wartości liczników wydajności oraz dane śledzenia zdarzeń pozwoli użytkownikom systemu Icinga 2 dokładnie poznać stan systemu.

## Założenia

### Wymagania

Podstawowe funkcje, które musi spełniać wtyczka diagnostics-windows, to:

* przeprowadzenie diagnostyki – w przypadku wystąpienia istotnego obciążenia systemu operacyjnego, zostanie uruchomiony zdefiniowany przez użytkownika zestaw modułów zbierających dane w programie Monitor Wydajności,
* przesłanie raportu – program dostarczy informację o nadmiernym wykorzystaniu zasobów oraz raport z przeprowadzonej diagnostyki do serwera monitorującego,
* prezentacja raportu – użytkownik będzie mógł wyświetlić raport w ramach interfejsu graficznego Icinga Web 2
* analiza danych historycznych – wszystkie otrzymane raporty będą przechowywane, ponadto użytkownik powinien mieć możliwość analizy wystąpień sytuacji krytycznych w czasie.

Aby wymienione powyżej funkcje działały prawidłowo musi także zostać spełniony szereg dodatkowych wymagań związanych ze zbieraniem, przesyłaniem i przechowywaniem danych. Należą do nich:

* odporność na brak łącza – jeżeli w momencie utworzenia raportu urządzenie nie ma dostępu do sieci to wszystkie dane powinny być przechowane, a po odzyskaniu połączenia wysłane do systemu monitorującego z prawidłową datą wystąpienia sytuacji krytycznej.
* transakcyjność – błędy w transmisji nie mogą skutkować utraceniem spójności danych. Wtyczka powinna gwarantować prawidłowe przesłanie raportu i informacji do Icingi.
* bezpieczeństwo – serwer monitorujący może znajdować się poza siecią lokalną, w której działa urządzenie. Dane powinny być w trakcie przesyłania oraz przechowywania chronione przed ujawnieniem osobom nieuprawnionym.
* oszczędność pasma oraz dysku – diagnostyka powinna być wykonywana jedynie w przypadku rzeczywistego wystąpienia obciążenia, a nie chwilowego i nagłego wzrostu wykorzystania danego zasobu. Zbyt częste dostarczanie niepotrzebnych raportów utrudniłoby także pracę osób nadzorujących infrastrukturę.
* Konfigurowalność – użytkownikowi powinien móc dostosować monitorowane parametry do własnych potrzeb. Oznacza to możliwość określenia zasobów, które będą monitorowane, a także zmienia zestawu modułów zbierających dane oraz czasu wykonania diagnostyki.

### Wybrane rozwiązania

Z założonego przeznaczenia stworzonej wtyczki, czyli monitorowania systemów Windows, wynika, że będzie ona działała na urządzeniach zdalnych (w pełni funkcjonalny rdzeń Icinga 2 działa jedynie na systemach linuxowych). Oznacza to, że do przekazywania danych do Icingi potrzebne jest wykorzystanie aplikacji pośredniczącej. Od wydania wersji drugiej systemu, twórcy rekomendują używanie rdzenia Icingi działającego na systemie Windows. Charakter zadania, czyli wysyłanie nieregularnych zgłoszeń do systemu Icinga wskazuje na wykorzystanie monitorowania pasywnego. Ten typ sprawdzeń jest nieobsługiwany przez wersję Icingi na systemy Windows, jednak nie wyklucza to całkowicie tego rozwiązania.

Sposób ten polegałby na przeprowadzaniu aktywnych sprawdzeń przez wbudowane wtyczki, a w momencie zwrócenia przez nie wartości krytycznej, uruchomieniu aplikacji *diagnostics-windows*, która przeprowadziłaby diagnostykę i wysłała raport. Wykorzystanie szablonów i instrukcji warunkowych sprawiłoby, że konfiguracja odpowiednich usług nie byłaby skomplikowana. Zaletą tej metody jest posłużenie się generycznymi mechanizmami. Oprócz tych wymienionych wyżej, użytkownik Icingi może także zdefiniować, kiedy (po ilu pomiarach) usługa przechodzi ze stanu *soft* do *hard*, co zapobiegałoby zbyt pochopnemu generowaniu raportów. Jednak czynnikiem przekreślającym ten wariant jest brak możliwości przesyłania raportu. Icinga może przyjmować dane jedynie do wielkości 8 KB. Nawet gdyby to ograniczenie zostało zniesione, przesłanie raportu jako wyjścia wtyczki byłoby bezcelowe, gdyż wszystkie elementy znaczników HTML są zamieniane na znaki specjalne. Ponadto niespełnione byłyby wymagania: dotyczące transakcyjności (raport przekazywany inną drogą niż informacja do głównego węzła Icingi) oraz przechowywania danych historycznych (w bazie danych systemu Icinga przetrzymywane są jedynie informacje o aktualnym stanie systemu).

Z opisanych powyżej przyczyn, do komunikacji z systemem Icinga została wykorzystana ulepszona wersja dodatku NSCA stworzona przez pana Krzysztofa Opasiaka. W przeciwieństwie do poprzednika nie zafałszowuje on stempla czasu wykonania sprawdzenia, co pozwala na wysyłanie do Icingi danych historycznych, a także wykorzystuje protokół spełniający stawiane kryteria bezpieczeństwa, tzn. szyfrowanie oraz spójność informacji. Dodatkowo program NSCAv2 posiada możliwość rozbudowania o wyspecjalizowane moduły przetwarzające otrzymane dane. Jest to szczególnie ważne gdyż dopuszcza umieszczenie raportu na przykład w przestrzeni dyskowej serwera monitorującego, a nie w systemie Icinga.

Rezygnacja z wykorzystania agenta Icingi, oznaczało również konieczność stworzenia własnego mechanizmu kontrolowania obciążenia systemu. Do pobrania poszczególnych wartości parametrów komputera wykorzystane zostały liczniki wydajności, które oferują dostęp do bardzo szerokiego zbioru atrybutów systemu. Równocześnie pozwalają one na łatwe zarządzanie, dzięki czemu można w prosty sposób zapewnić użytkownikowi wybór monitorowanych elementów w zależności od jego preferencji.

Ostatnim elementem, którego wybór należało rozważyć był dodatek graficzny, służący do analizowania sytuacji krytycznych w czasie. Jest to bardzo ważna funkcjonalność, pozwalająca zaobserwować tendencje zmian obserwowanych wartości. Mimo to nie została ona zaimplementowana w aplikacji Icinga Web 2, dlatego też trzeba korzystać z dodatkowych programów opisanych w 2.4. Żaden z nich nie posiada cech, pozwalających na jednoznaczny wybór najlepszego rozwiązania. W kontekście wygody użytkowania należałoby wskazać pnp4nagios, dla którego stworzono moduł do Icingi Web 2, jednak oferowane przez niego funkcje edycji wykresów są zdecydowanie zbyt małe. Z pozostałych dwóch programów wybrałem InGraph, ze względu na fakt, iż korzysta on z baz relacyjnych, a nie cyklicznych. Zapobiega to niekontrolowanej utracie danych, które w przypadku monitorowania sprzętu komputerowego, mogą być potrzebne do wglądu nawet w okresie lat. Warto zauważyć, że oba dodatki InGraph oraz Graphite mogą być używane jednocześnie, gdyż wykorzystują różne mechanizmy pobierania danych wydajnościowych.

## Architektura

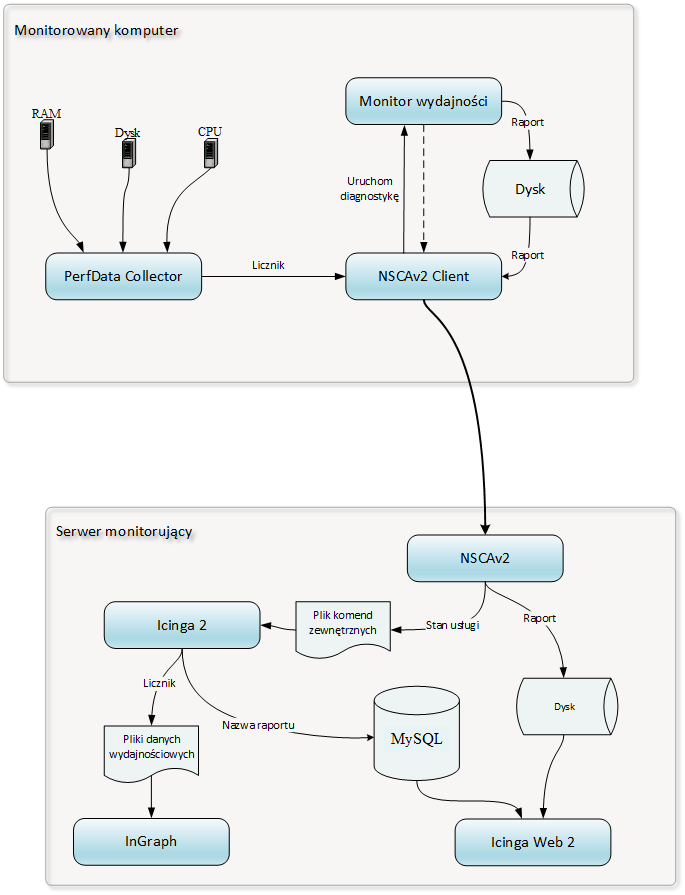
Analiza wybranych rozwiązań wskazała, że do wtyczka *diagnostics-windows* powinna składać się z trzech modułów:

* monitorującego bieżący stan parametrów systemu,
* odpowiedzialnego za wykonanie diagnostyki Monitora wydajności oraz przygotowanie raportu do wysłania,
* wysyłającego raport do dodatku NSCAv2.

Ponadto, aby część serwerowa systemu monitorującego działała poprawnie, czyli odpowiednio obsługiwała przychodzące raporty konieczne było stworzenie modułów rozszerzających programy NSCAv2 oraz Icinga Web 2. Pełna architektura systemu Icinga 2 i współpracującej z nim wtyczki *diagnostics-windows* została przedstawiona na rysunku 4.1.

Wtyczka *diagnostics-windows* składa się z dwóch oddzielnych aplikacji komunikujących się za pomocą gniazd TCP. Pierwsza z nich *PerfData Collector* zawiera implementację modułu sprawdzającego stan liczników wydajności. W momencie, gdy któryś z nich przekroczy wartość progową (ustaloną przez użytkownika) do aplikacji *NSCAv2 Client* wysyłana jest informacja o wystąpieniu sytuacji krytycznej w postaci nazwy licznika, który przekroczył próg. Program *NSCAv2 Client* zawiera dwa pozostałe moduły działające na urządzeniu zdalnym. W chwili otrzymania wiadomości o obciążeniu komputera, uruchamia on diagnostykę systemu w Monitorze Wydajności*.* Następnie wygenerowany raport jest wczytywany z dysku i po odpowiednim przygotowaniu, pozwalającym na odebranie i sparsowanie przez dodatek NSCAv2, przekazywany do serwera monitorującego. Proces utworzenia i wysłania raportu może być inicjowany nie tylko w sytuacji przekroczenia limitu przez liczniki wydajności. Dzięki zastosowaniu odrębnej aplikacji do obsługi i wysyłania raportów, nasłuchującej na gnieździe TCP, istnieje możliwość dodania innych programów, które będą wysyłały żądanie przeprowadzenia diagnostyki.

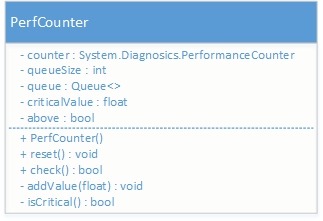
Na serwerze monitorującym dane są odbierane przez dodatek NSCAv2. Do ich przetworzenia został stworzony specjalny moduł, który z tekstowego wyjścia wtyczki wyodrębnia treść raportu i zapisuje ją w ustalonym miejscu na dysku. Do pliku komend zewnętrznych trafia natomiast komenda przetworzenia sprawdzenia stanu usługi. Zawiera ona nazwę utworzonego raportu oraz, w części danych wydajnościowych, nazwę licznika, który odpowiadał za przeprowadzenie diagnostyki. Na podstawie tej drugiej informacji program InGraph wygeneruje wykresy, które umożliwią analizę kształtowania się obciążenia poszczególnych liczników (komponentów komputera) w czasie. Tekstowa część wyjścia wtyczki (nazwa raportu) zgodnie ze standardowym działaniem systemu Icingi 2 zostanie zapisana w bazie danych aplikacji. Ostatnim elementem opisywanego procesu jest moduł rozszerzający interfejs Icinga Web 2, który pozwala na przeglądanie wygenerowanych raportów. Listę dostępnych dokumentów pobiera on z bazy danych Icingi*,* a ich treść wczytuje z odpowiednich plików na dysku, które zostały utworzone przez dodatek NSCAv2.



**Rys. 4.1** Schemat działania wtyczki diagnostics-windows

## Opis modułów

### PerfData Collector

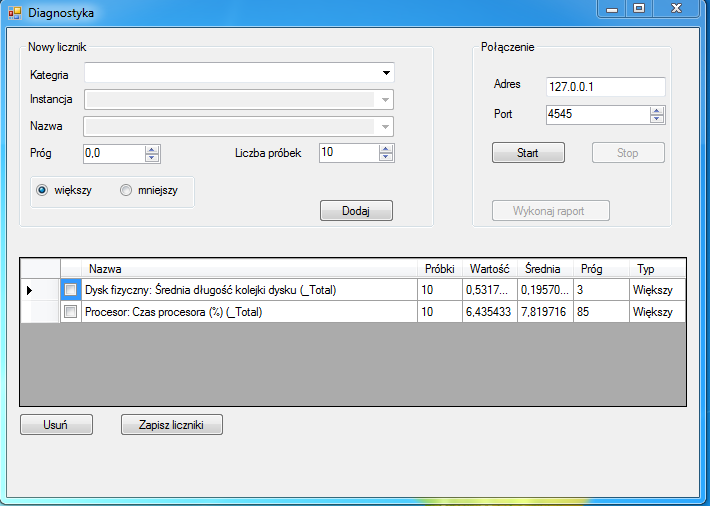
Moduł monitorowania stanu systemu służy do informowania wtyczki *diagnostics-windows* o potrzebie wykonania diagnostyki. W tym celu pobiera on cyklicznie wartości liczników wydajności, a w momencie, w którym wykryje zbyt duże obciążenie zasobów, wysyła informację do programu *NSCAv2 Client.* Moduł ten został zaimplementowany w języku C#, który stworzono dla firmy Microsoft, przez co zawiera wiele bibliotek dających dostęp do natywnych interfejsów systemu Windows. Umożliwiło to proste i efektywne zarządzanie oraz pobieranie wartości liczników.

**Rys. 4.2** Atrybuty oraz metody klasy PerfCounter

Najważniejszym elementem modułu jest klasa *PerfCounter* (rys. 4.2)reprezentująca monitorowany licznik wydajności. Zawiera ona dwie publiczne metody *reset()* i *check()*, służące odpowiednio do przywrócenia stanu początkowego parametrów oraz sprawdzenia wartości licznika. Wywołanie diagnostyki nie powinno odbywać się w rezultacie jednokrotnego przekroczenia wartości krytycznej, gdyż taka sytuacja mogłaby oznaczać jedynie chwilowy skok wykorzystania zasobu. Z tego powodu do klasy *PerfCounter* została dodana kolejka FIFO, która pozwala przetrzymywać wartości ostatnich pomiarów. Przy ustalonym okresie wywoływania metody *isCheck()*, długość tej kolejki oznacza minimalny czas, przez który musi się utrzymywać zwiększone obciążenie systemu, aby została rozpoznana sytuacja krytyczna. Metoda *isCritical(),* wywoływana podczas każdego sprawdzenia, na podstawie zawartości kolejki oblicza średnią wykonanych pomiarów i porównuje ją z wartością progową (flaga *above* określa znak tego porównania). Jeżeli zostanie ona przekroczona oraz kolejka jest pełna, funkcja zwróci wartość *true,* oznaczającą konieczność wysłania informacji o sytuacji krytycznej do modułu obsługi raportów.

Działanie programu regulowane jest przez obiekt klasy Timer, który z ustalonym okresem generuje zdarzenia Tick. Ich obsługa polega na zebraniu listy kategorii, z których liczniki przekroczyły wartość progową. Jeśli lista ta nie jest pusta to przekazywana jest poprzez protokół TCP do programu *NSCAv2 Client*, a obiekty klasy *PerfCounter* powracają do stanu początkowego (wywołanie metody *reset()*).

W celu zapewnienia użytkownikowi wygodnego zarządzania monitorowanymi elementami komputera, do aplikacji *PerfData Collector* został dołączony interfejs graficzny (rys. 4.3). Pozwala on na wybór monitorowanych liczników oraz związanych z tym parametrów:

* wartości progowej,
* relacji między progiem a rezultatem sprawdzenia (określa czy sytuacja krytyczna występuje powyżej czy poniżej progu),
* liczby próbek, czyli de facto okresu, z którego liczona jest średnia wartość danego licznika.

**Rys. 4.3** Okno programu PerfData Collector

Ponadto w oknie programu znajduje się tabela pozwalająca obserwować bieżący stan i ustawienia wybranych liczników oraz sekcja poświęcona połączeniu TCP. Użytkownik może w niej określić docelowy adres oraz port, a także uruchomić komunikację. Jeżeli tego nie zrobi, program będzie normalnie monitorował liczniki, jednak nie będzie wysyłał wiadomości do wskazanej lokalizacji.

### Moduł obsługi raportów

Moduł ten odpowiada za przyjęcie informacji o sytuacji krytycznej i przekazaniu wygenerowanego raportu do wysłania. Po starcie programu, w oddzielnym wątku tworzy on gniazdo TCP i oczekuje na nim na wiadomości o sytuacji krytycznej. W momencie odebrania takiej informacji zostaje utworzony nowy wątek odpowiadający za przeprowadzenie diagnostyki. Jest ona uruchamiana za pomocą skryptu Powershell, który po prawidłowym wykonaniu zwraca lokalizację raportu na dysku. Ponieważ skrypt jest zasobem współdzielonym i istnieje możliwość wielokrotnego wykonywania diagnostyki w tym samym czasie, dostęp do niego kontrolowany jest za pomocą semaforów binarnych. Po otrzymaniu lokalizacji pliku, do raportu w formacie HTML dodawany jest stempel czasu oraz lista kategorii liczników, które wywołały diagnostykę. Informacje te, a także lokalizacja raportu opakowywane są w obiekt klasy *PerfomonResult*, który następnie umieszczany jest w kolejce blokującej[[9]](#footnote-9). Stamtąd odczytywana jest ona przez wątek komunikujący się z dodatkiem NSCAv2. Lista kategorii jest przekazywana dalej, ponieważ trafia ona na wyjście wtyczki jako dane wydajnościowe. Po udanym wysłaniu, raport jest usuwany z dysku.

Moduł ten odpowiedzialny jest także za dosyłanie raportów, które znajdują się na dysku w momencie uruchomienia aplikacji. Jeżeli ze względu na brak połączenia z serwerem, nie uda się wysłać raportu i w międzyczasie aplikacja zostanie wyłączona to moduł wysyłający straci informację o raportach, które powinien przekazać. Jednak pozostają one zapisane na dysku, dlatego po uruchomieniu aplikacji z każdego z nich odczytywany jest stempel czasu i kategorie liczników, a następnie informacja o tych raportach jest ponownie umieszczana w kolejce blokującej.

### Moduł komunikacji z dodatkiem NSCAv2

Dodatek NSCAv2w porównaniu do swojego poprzednika wprowadził wiele ulepszeń. Aby je spełnić, konieczne było użycie protokołu komunikacyjnego, który przede wszystkim zapewniałby bezpieczeństwo. Pan Opasiak zdecydował się na stworzenie własnego protokołu opartego na protokole TCP, który jest dopasowany do wymagań problemu i nie wprowadza nadmiarowego nakładu danych. Komunikacja z użyciem tego protokołu składa się z następujących etapów:

* nawiązanie połączenia,
* wysłanie komunikatu *hello*,
* wybór wersji protokołu,
* potwierdzenie identyfikatora klienta,
* wybór algorytmu szyfrowania symetrycznego,
* przesłanie klucza algorytmu symetrycznego,
* autoryzacja klienta,
* przesłanie danych,
* zakończenie komunikacji.

W zastosowanym modelu kryptografii do przesłania klucza szyfrowania symetrycznego wykorzystywane jest szyfrowanie asymetryczne. Z tego powodu, podczas pierwszej instalacji aplikacji lub po zmianie klucza prywatnego serwera, na każdej maszynie zdalnej trzeba ręcznie umieścić odpowiedni klucz publiczny. Warto zauważyć, że protokół ten nie wymaga certyfikatów bezpieczeństwa służących do autoryzacji serwera. Istotną cechą protokołu jest także wysyłanie potwierdzeń otrzymanych danych, dzięki czemu klient ma pewność, że dane dotarły w całości i można je bezpiecznie usunąć z monitorowanego urządzenia.

Tworząc aplikację *NSCAv2 Client* zdecydowałem się użyć istniejącej implementacji obsługi protokołu po stronie klienta. W tym celu wykorzystałem program stworzony przez pana Marcina Kubika w ramach pracy „Rozproszony system monitorowania systemów komputerowych – aplikacja mobilna na platformę Android”. Jak wskazuje tytuł jest to program działające na systemie Android i napisany w języku JAVA, co umożliwiło jego przeniesienie, po drobnych modyfikacjach, na system Windows.

W aplikacji *NSCAv2 Client* posłużyłem się pakietami dotyczącymi komunikacji oraz kryptografii. Komunikacja odbywa się w oddzielnym wątku, który w zależności od etapu, znajduje się w innym stanie. Stany te są reprezentowane przez klasy implementujące interfejs *SocketConnectionState*. Do wysyłania i odbierania wiadomości używają one metod dostarczonych przez klasy *MessageFormer* i *MessageDecrypter*. Takie rozwiązanie, korzystające z polimorfizmu, pozwoliło na ujednoliconą obsługę każdego stanu w głównej pętli komunikacji. Kryptografia kontrolowana jest przez klasę *CryptoManager*, stworzoną na podstawie wzorca projektowego *singleton*. Udostępnia ona metody, pozwalające korzystać z pozostałych klas znajdujących się pakiecie *crypto*:

* KeyLoader – wczytuje z dysku klucz publiczny,
* AESModule – obsługuje szyfrowanie symetryczne, w tym generację klucza i wektora inicjalizującego,
* RSAModule – odpowiedzialny za część protokołu szyfrowaną asymetrycznie,
* SHAModule – generuje i sprawdza skróty wiadomości.

Modyfikacje wprowadzone do programu pana Kubika miały dwojaki charakter: dostosowania aplikacji do pracy na systemie Windows oraz dostosowania do wymagań wtyczki *diagnostics-windows*. Na nowo zostały zaimplementowane klasy służące do wczytywania konfiguracji i logowania zdarzeń. Ze względu na brak interfejsu graficznego korzystają one z plików tekstowych, znajdujących się w folderze aplikacji:

* pliki logów –dla każdego uruchomienia programu tworzony jest oddzielny plik,
* plik konfiguracyjny - ma format XML i umożliwia ustawienie wszystkich parametrów dotyczących komunikacji (adres, nazwa hosta, login, …).

Zmianie uległo też działanie wątku komunikacji. Ponieważ dane wysyłane są nieregularnie, w sytuacjach nadzwyczajnych, to nie jest konieczne utrzymywanie połączenia na gnieździe TCP. W momencie pojawienia się elementu w kolejce blokującej (elementy do kolejki wstawia moduł obsługi raportów), wątek jest wybudzany i nawiązuje połączenie z dodatkiem NSCAv2. Jeżeli raport zostanie wysłany poprawnie, połączenie zostaje zakończone, a wątek ponownie oczekuje na dane w kolejce. W przeciwnym wypadku, wątek zostaje wstrzymany na czas określony przez użytkownika, po którym ponownie próbuje połączyć się z serwerem.

Kolejnym zmodyfikowanym elementem ze względu na charakter zadania, było wysyłanie wyjścia wtyczki. Wymagało to nadpisania metody *formLog()* w klasie dziedziczącej po *MessageFormer*. Protokół komunikacyjny jest mało elastyczny, ponieważ przesyłane dane, muszą mieć strukturę komend systemu Icinga 2. W przypadku odejścia od tej formy dane są odrzucane przez NSCAv2 jako nieprawidłowe. Z tego względu treść raportu została umieszczona w części przeznaczonej na wyjście tekstowe wtyczki, od którego oddzielona jest separatorem. Struktura wiadomości umożliwia także wysłanie danych wydajnościowych. W przypadku wtyczki *diagnostics-windows* jest to lista kategorii liczników wydajności, które przekroczyły wartość krytyczną. Aby były one akceptowane przez system Icinga muszą mieć format *klucz – wartość*. W tym celu do nazwy każdej kategorii dodawany jest tekst „=1”. Oznacza to, że na wykresach generowanych w programie InGraph, w przypadku wystąpienia danej kategorii będzie zaznaczana wartość 1.

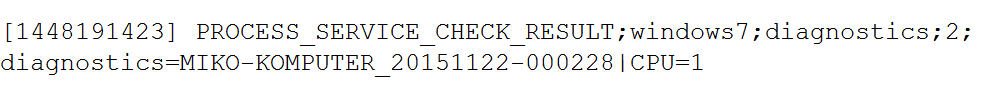
### Moduł konsumenta dodatku NSCAv2

W poprzednim punkcie zostało zaznaczone, że treść raportu znajduje się w wyjściu tekstowym wtyczki. Jak wskazała analiza założeń, jej umieszczenie w bazie danych systemu Icinga 2 jest bezcelowe. Zadaniem tego modułu jest wydzielenie tekstu raportu i zapisanie go na dysku.

Dodatek NSCAv2posiada modułową budowę, która umożliwia jego rozszerzanie. W szczególności może to dotyczyć modułu przetwarzającego dane. W programie pana Opasiaka zostały wyróżnione trzy rodzaje komponentów, przez które przechodzą odebrane dane:

* dostawcy danych,
* kanał komunikacyjny,
* konsumenci danych.

Każdy z nich może posiadać wiele implementacji, a to jak będą przetwarzane dane, ustalane jest w pliku konfiguracyjnym na podstawie identyfikatora programu wysyłającego wiadomość. Za przetworzenie i zapisanie danych odpowiedzialni są konsumenci danych, dlatego do implementacji modułu stworzyłem klasę tego typu.

W metodzie *conusmeDataPortion(…)* wydziela ona z wyjścia wtyczki treść raportu na podstawie ustalonego separatora, a następnie raport zapisywany jest w pliku HTML na dysku serwera. Pliki raportów trzymane są w katalogu publicznych stron serwera Apache (/var/www/html/reports), co umożliwia do nich dostęp bez wykorzystania systemu Icinga. Pozostała część danych, czyli właściwe wyjście wtyczki umieszczane jest w pliku komend zewnętrznych(rys. 4.4) w identyczny sposób jak w konsumencie danych dostarczonym wraz z dodatkiem NSCAv2*.*

**Rys. 4.4** Rezultat sprawdzenia pasywnego umieszczany w pliku komend zewnętrznych

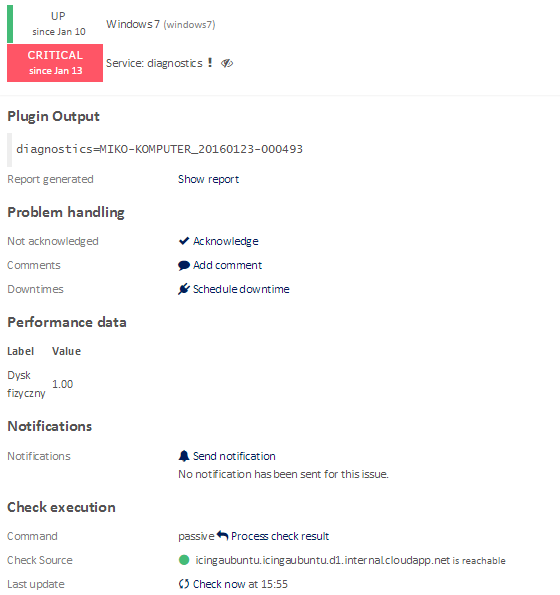
### Moduł interfejsu graficznego Icinga Web 2

Jest to moduł rozszerzający interfejs graficzny Icinga Web 2 o możliwość wyświetlania przesłanych na serwer raportów. Z tego względu musi on spełniać wymagania, które zostały dokładnie opisane w dodatku B niniejszej pracy. Między innymi musi on być zgodny z platformą programistyczną *ZendFramework* przeznaczoną do tworzenia aplikacji internetowych w języku PHP*.*

Do zaimplementowania modułu została użyta jedna klasa kontrolera dziedzicząca po klasie *ActionController* znajdującej się w bibliotekach *ZendFramework.* Jest ona wywoływana w momencie otwarcia strony modułu i zwraca kod HTML zawierający listę otrzymanych raportów. Lista ta jest pobierana z bazy Icingi, jednak ze względu na fakt, że Icinga nie przechowuje danych historycznych, aby to osiągnąć potrzebne były zmiany w konfiguracji. Do obiektu usługi został dodany atrybut *volatile*, który powoduje wysyłanie powiadomień za każdym razem, gdy wtyczka zwróci stan krytyczny bez względu stan (hard/soft) usługi. Jednocześnie informacja o stworzeniu powiadomienia odnotowywana jest w bazie danych w tabeli *icinga\_statehistory*. Składa się ona między innymi z wyjścia wtyczki, które zawiera nazwę raportu.

Aby wyświetlić listę odebranych raportów do głównego menu interfejsu dodano sekcję *Windows diagnostics*, która przekierowuje do strony modułu. Po wybraniu przez użytkownika raportu, jest on wyświetlany w ramce *iframe*. Źródłem dla niej jest dokument HTML z wynikami diagnostyki znajdujący się w publicznym katalogu serwera Apache. Do obsługi wyświetlania raportu została użyta funkcja javascript, ustawiająca w znacznikach *iframe* właściwyatrybut źródła.

Ponadto, aby zwiększyć wygodę użytkownika do widoku usługi zostało dodane łącze, pozwalające wyświetlić raport odpowiadający aktualnemu wyjściu wtyczki (rys.4.5) . W konfiguracji Icinga 2 obiekt usługi posiada atrybut *action\_url*, do którego można przypisać link, który następnie będzie wyświetlany w sekcji *actions*. Aby mógł on prowadzić do raportu, atrybut ten potrzebuje korzystać z makra *$service.output*, oznaczającego wyjście wtyczki. Niestety rozwiązanie to stanowiło przykład niedoskonałości systemu Icinga 2, gdyż makro to nie było rozwijane, mimo zapisów w dokumentacji o poprawnym działaniu. Z tego powodu do wyświetlenia linku został użyty jeden z zaczepów (ang. hook) pozostawionych w kodzie Icinga Web 2, pozwalający na integrację z dodatkami graficznymi. Może on jednak służyć do innych celów, gdyż umożliwia implementację metody, zwracającej kod HTML umieszczany pod wyjściem wtyczki.



**Rys. 4.5** Widok stanu usługi

# Testy aplikacji

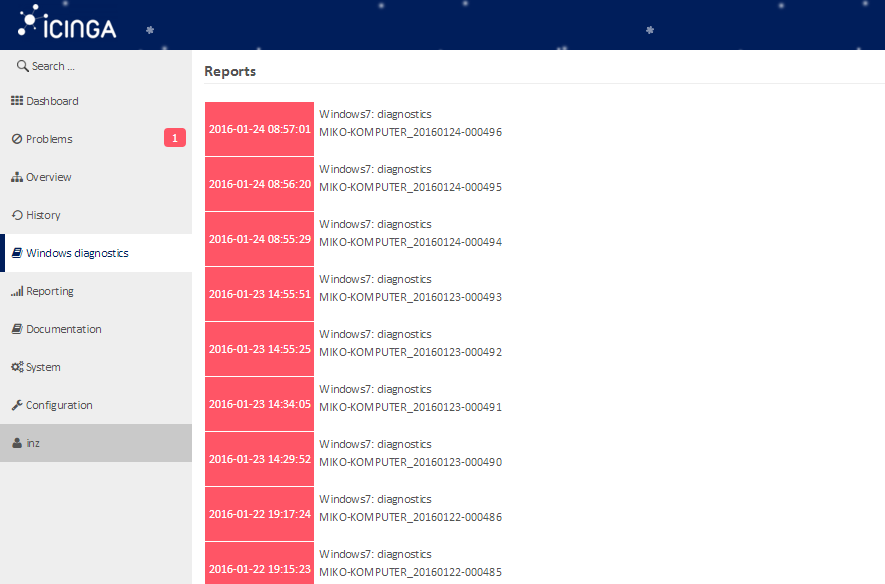
Testowanie jest jednym z najważniejszych etapów procesu wytwarzania oprogramowania, który pozwala zweryfikować czy aplikacja spełnia stawiane przed nią założenia. W ramach testów wtyczka *diagnostics-windows* została zainstalowana na laptopie o następujących parametrach:

* Windows 7 64-bit,
* procesor Intel Core i5-3337U,
* pamięć DDR3 4GB,
* Toshiba MQ01ABD050 500GB,
* karta siciowa Atheros ar9485wb-eg.

Do uruchomienia systemu Icinga została wykorzystana platforma Microsoft Azure, która umożliwiła stworzenie maszyny wirtualnej służącej jako serwer monitorujący. Takie rozwiązanie pozwoliło uniknąć obciążenia monitorowanego komputera przez działający na nim serwer oraz przetestować działanie komunikacji nie tylko w sieci lokalnej, gdzie ryzyko utraty pakietów TCP jest znacznie mniejsze. Na maszynie wirtualnej został uruchomiony system Ubuntu 14.04 oraz następujące oprogramowanie systemu monitorowania:

* serwer http Apache 2.4.7,
* baza danych MySql,
* Icinga 2.3.4,
* Icinga Web 2.1.2,
* InGraph 1.0.2,
* NSCAv2.

Testy zostały przeprowadzone na przestrzeni dwóch tygodni, podczas których do systemu Icinga raporty były wysyłane bezpośrednio po wykonaniu diagnostyki, a także po pewnym czasie, w celu sprawdzenia czy są one odbierane z prawidłowym czasem wykonania (rys. 5.1). Jedynym zaobserwowanym mankamentem, było przesunięcie o godzinę czasu wykonania raportów, co wynika z faktu używania przez serwer strefy czasowej GMT.

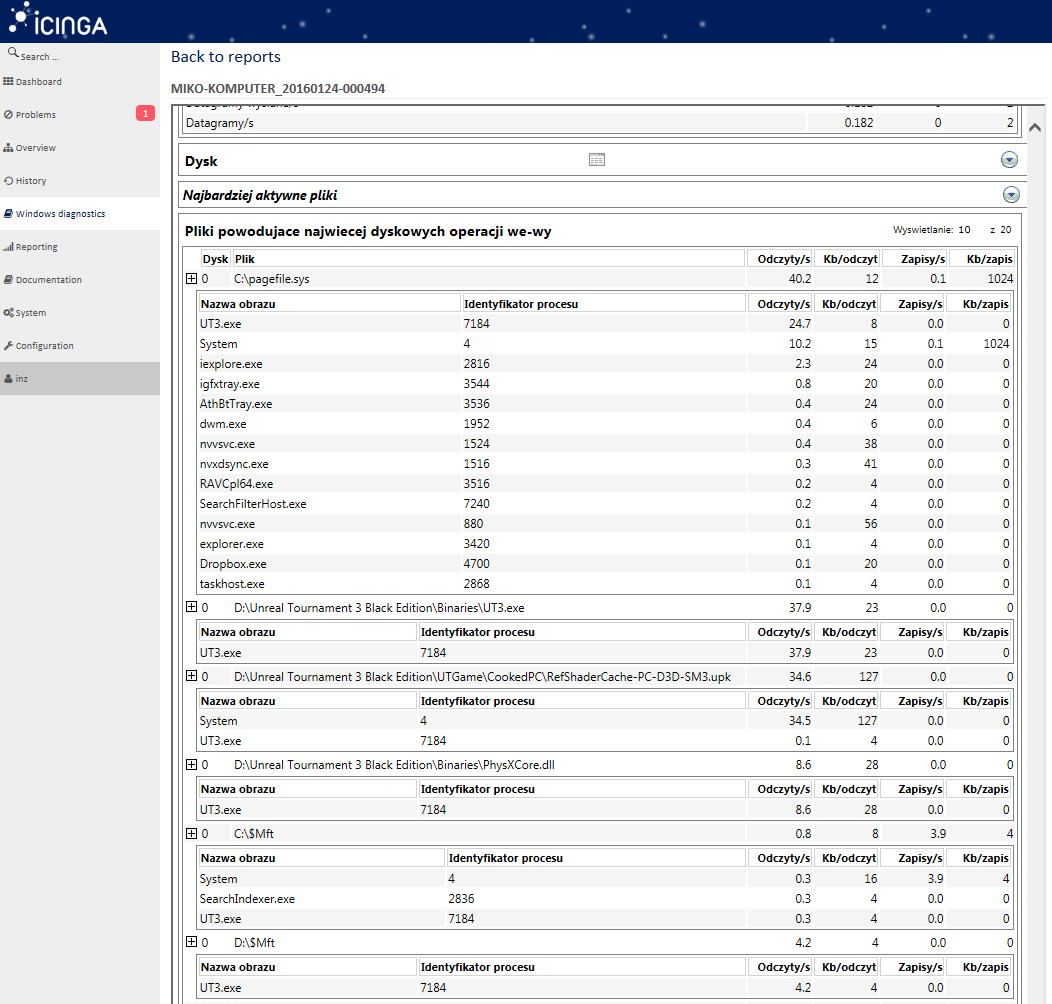


**Rys. 5.1** Widok listy otrzymanych raportów w module Windows diagnostics

W trakcie działania na monitorowanym komputerze wtyczka *diagnostics-windows* sprawdzała wartości 10 liczników opisanych w tabeli 3.1. W tym czasie wykryła 45 sytuacji krytycznych, w wyniku których została uruchomiona diagnostyka, a raporty zostały przesłane do serwera Icinga. Na rys 5.2 przedstawiono, które kategorie liczników wskazywały obciążenie zasobów.

**Rys. 5.2** Ilość sytuacji krytycznych w zależności od kategorii licznik

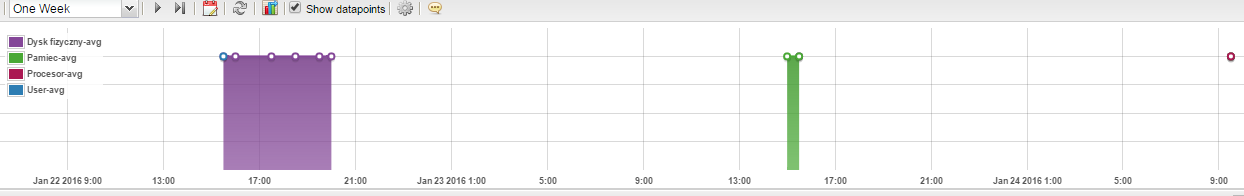
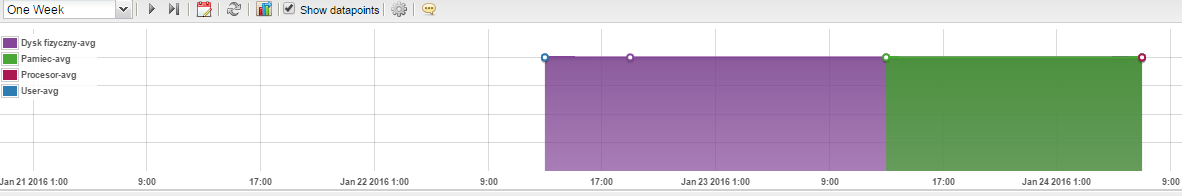
Jak widać na wykresie „wąskim gardłem” monitorowanego komputera był dysk twardy, który dwukrotnie częściej był nadmiernie obciążony niż pozostałe liczniki razem wzięte. Szczegółowa analiza wygenerowanych raportów wskazała, że w prawie połowie przypadków plikiem odpowiedzialnym za wysoką wartość kolejki dysku był plik *pagefile.sys* (rys. 5.3). Jest to plik, który razem z pamięcią RAM tworzy pamięć wirtualną systemu. Do *Pagefile.sys* przenoszone są dane z pamięci podręcznej programów, które nie były od dłuższego czasu używane. Mechanizm ten pozwala na zyskanie pamięci fizycznej w przypadku jej dużej zajętości. Zatem mimo, że liczniki pamięci nie przekraczały ustawionych progów, to pośrednio była ona również przyczyną nadmiernego użycia dysku.

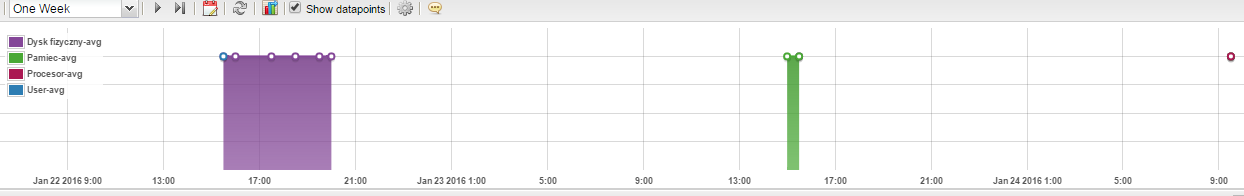


**Rys. 5.3** Widok raportu w module Windows diagnostics

Ostatnim elementem procesu monitorowania za pomocą wtyczki *diagnostics-windows* była analiza danych wydajnościowych w programie InGraph. Niestety program ten przeznaczony jest bardziej dla danych przekazywanych regularnie w niewielkich odstępach czasu. Posiada on własną politykę agregowania danych, która nie jest konfigurowalna, przez co otrzymane wyniki nie były dokładnie czytelne. Zostało to przedstawione na rysunku 5.4, gdzie porównano dwa wykresy, które w zależności od ustawionej skali, wskazywały w tym samym okresie inną liczbę otrzymanych raportów.

**Rys. 5.4** Dane wydajnościowe w programie InGraph





# Podsumowanie

Głównym założeniem pracy było zapoznanie się ze sposobami monitorowania systemów Microsoft Windows, jakie daje narzędzie Icinga. Przeprowadzony w tym celu przegląd aplikacji współpracujących z tym systemem pokazał, że zbiór wykorzystywanych rozwiązań nie jest pełny. Dlatego w ramach pracy został stworzony program *diagnostics-windows*, który uruchamia i przekazuje wynik diagnostyki systemu do serwera monitorującego. Umożliwia to użytkownikom systemu Icinga poznanie przyczyn obciążenia systemu, gdyż raport zawiera między innymi informacje o procesach wykorzystujących zasoby, a także dane z dziennika śledzenia zdarzeń. Mimo, że są one bardzo istotne, dotychczas nie były używane w programach współpracujących z systemem Icinga. Działanie sporządzonej wtyczki wiązało się również z dostosowaniem interfejsu oraz dodatku odbierającego dane do wymogów przesyłania i wyświetlania dokumentu HTML zawierającego raport. W tym celu do obu wymienionych programów zostały stworzone odpowiednie moduły.

W trakcie tworzenia wtyczki zaobserwowałem dwie znaczące trudności. Pierwszą z nich był zakres użytych technologii. Implementacja sporządzonych aplikacji oraz modułów wymagała użycia czterech języków programowania: C#, JAVA, C++ oraz PHP. Drugim problemem były braki w dokumentacji systemu Icinga 2. Jest to system nowy i cały czas rozwijany, dlatego często, do wyjaśnienia nieścisłości, konieczne było szukanie pomocy na forach dyskusyjnych. Z tego powodu w pracy został zamieszczony rozdział, systematyzujący informację o tym systemie. Pozwala on poznać jego działanie, a także funkcjonalności wprowadzone w najnowszej wersji.

Przeprowadzone testy pokazały, że wnioski wyciągnięte z analizy sposobów monitorowania systemu Windows były trafne. Sama informacja o zwiększonym obciążeniu zasobu nie jest wystarczająca do rozwiązania problemu. Do jego pełnego zrozumienia potrzebne są dokładniejsze dane, których dobrym źródłem jest program Monitor Wydajności. Przeprowadzone testy wskazały także znaczącą lukę systemu Icinga 2, jaką jest brak uniwersalnego dodatku graficznego. Stworzenie takiej aplikacji, która przechowywałaby odczyty w bazach relacyjnych, integrowała się z interfejsem Icinga Web 2 oraz dawała użytkownikowi kontrolę nad agregacją danych, może być kolejnym interesującym zadaniem rozszerzającym system Icinga 2.

# Dodatek A – instalacja systemu Icinga 2

W poniższym załączniku opisałem kroki potrzebne do instalacji systemu Icinga 2, w którego skład wchodzi rdzeń monitorujący, interfejs graficzny Icinga Web 2 oraz dodatki InGraph i NSCAv2. Ten poradnik dotyczy wersji systemu i oprogramowania wymienionych w rozdziale 5 pracy.

* **Rdzeń monitorujący Icinga 2**

1. Dodanie repozytorium i instalacja

> add-apt-repository ppa:formorer/icinga

> apt-get update

> apt-get install icinga2

1. Instalacja bazy danych

> apt-get install mysql-server mysql-client

1. Instalacja modułu IDO

> apt-get install icinga2-ido-mysql

1. Stworzenie bazy danych

> mysql

mysql> CREATE DATABASE icinga2;

mysql> GRANT SELECT, INSERT, UPDATE, DELETE, DROP, CREATE VIEW, INDEX, EXECUTE ON icinga.\* to ‘icinga’@’localhost’ IDENTIFIED BY ‘icinga’;

1. Import schemata bazy danych

> mysql –p icinga2 < /usr/share/icinga2-ido-mysql/schema/mysql.sql

1. Uruchomienie modułów

> icinga2 feature enable ido-mysql

> icinga2 feature enable perfdata

> icinga2 feature enable command

> service icinga2 restart

1. Dodanie konfiguracji do obsługi wtyczki *diagnostics-windows* (plik z konfiguracją znajduje się na płycie z projektem)

> mv diagnostics\_conf.txt /etc/icinga2/conf.d/diagnostics.conf

> service icinga2 restart

* **Icinga Web 2**

1. Instalacja serwera

> apt-get install apache2

1. Dodanie repozytorium i instalacja

> wget -O - http://packages.icinga.org/icinga.key | apt-key add -

add-apt-repository 'deb http://packages.icinga.org/ubuntu icinga-trusty main'

> apt-get update

> apt-get install icingaweb2

1. Generacja tokenu – jest on potrzebny do uwierzytelnienia w trakcie konfiguracji

> icingacli setup token create

> icingacli setup token show

1. Konfiguracja

Aby skonfigurować interfejs graficzny należy w przeglądarce w adresie strony wpisać *localhot/icingaweb2.* Po wpisaniu utworzonego w poprzednim punkcie tokenu zostanie uruchomiony kreator ustawień. Należy uważać na podanie prawidłowych baz danych, gdyż kreator prosi o to dwukrotnie: 1) o bazę rdzenia, 2) o bazę interfejsu do przechowywanie osobistych ustawień użytkowników (jest tworzona w trakcie konfiguracji).

1. Dodanie modułu do wyświetlania raportów (moduł znajduje się na płycie z projektem)

> mv /Reports /usr/share/icingaweb2/modules

1. Włączenie modułu w konfiguracji Icingi Web 2

localhost/icingaweb2/config/modules

* **InGraph**

1. Instalacja wymaganych pakietów:

python-sqlalchemy python-setuptools php5-curl php5-xmlrpcpython-mysqldb python-pygresql python-psycopg2

1. Instalacja

Należy pobrać i rozpakować plik: www.netways.org/attachments/download/806/inGraph.1.0.2.tar.gz

> cd InGraph.1.0.2

> ./setup-daemons.sh --install

> cd ingraph-web/

>./setup-ingraph-web.sh –install --prefix=/usr/local/ingraph-web \

--with-web-path=/ingraph

1. Ustawienia serwera Apache

W ingraph.conf zmienić:

Order allow, denny

Allow from all

na

Require all granted

> cp ingraph.conf /etc/apache2/sites-available

> a2ensite ingraph.conf

> service apache2 reload

1. Ustawienie bazy danych

> mysql

mysql> CREATE DATABASE ingraph;

mysql> GRANT ALL ON ingraph.\* TO 'ingraph'@'localhost'

-> IDENTIFIED BY 'changeme';

1. Konfiguracja dostępu do bazy danych.

> nano /etc/ingraph/ingraph-database.conf

dsn = 'mysql://ingraph:changeme@localhost:3306/ingraph'

1. Stworzenie konfiguracji kolektora danych wydajnościowych

> nano /etc/default/ingraph-collector

INGRAPH\_COLLECTOR\_PERFDATA\_DIR="/var/spool/icinga2/perfdata"

INGRAPH\_COLLECTOR\_PERFDATA\_PATTERN="\*-perfdata.\*[0-9]"

> chmod g+rwx /var/spool/icinga2/perdata

1. Uruchomienie

> service ingraph start

> service ingraph-collector start

* **NSCAv2**

1. Instalacja wymaganych pakietów

Qt 5.0, cmake, build essential,libboost-all-dev, doxygen, libcrypto++,

graphviz

1. Stworzenie wymaganych folderów

> mkdir /usr/local/nscav2

> mkdir /usr/loacl/nscav2/etc

> mkdir /usr/local/nscav2/crypto

> mkdir /usr/local/nscav2/var/buf

1. Wypakowanie projektu

> unzip nscav2-master.zip –d /usr/local/nscav2

1. Ustawienie w pliku *setup.sh* ścieżki do instalacji biblioteki Qt i uruchomienie skryptu

> nano /usr/local/nscav2/nascav2-master/

> ./setup.sh

1. Umieszczenie klucza prywatnego RSA w folderze:

/usr/local/nscav2/crypto

1. Dodanie konfiguracji (plik znajduje się na płycie z projektem)

> mv nscav2\_server\_config.xml /usr/local/nscav2/etc/config.xml

1. Dodanie plików do modułu konsumenta (znajdują się na płycie)

> mv consumer\_module/DiagnosticsConsumer.cpp …/nascav2-master/code/Base/src/

> mv consumer\_module/DiagnosticsConsumer.h …/nascav2-master/code/Base/include/

> mv consumer\_module/CMakeLists.txt …/nascav2-master/code/Base

1. Zbudowanie projektu

> cd …/nascav2-master/build

> make

1. Uruchomienie

> cd …/nascav2-master/build/bin

> ./base

# Dodatek B – tworzenie modułów do Icinga Web 2

Interfejs Icinga Web 2 został zaimplementowany z wykorzystaniem platformy programistycznej ZendFramework, dlatego każdy dodatek musi być zgodny z zastosowanymi tam rozwiązaniami. Podstawowa struktura umieszczanych plików wygląda następująco:

[module\_name]/application/controllers/IndexController.php

[module\_name]/application/views/scripts/index/index.phtml

Ponadto każdy moduł musi zawierać pliki:

[module\_name]/configuration.php

[module\_name]/module.info

Pierwszy z plików służy do umieszczenia sekcji z zakładką prowadząco do modułu lub jego ustawień, natomiast drugi zawiera nazwę, wersję i opis modułu, który będzie wyświetlany w ustawieniach interfejsu.

Aplikacja Icinga Web 2 zawiera także kilka zaczepów, pozwalających na dodawanie własnego kodu bez ingerencji w oryginalną implementację. Należą do nich:

* GrapherHook
* TicketHook
* DataviewExtensionHook
* HostActionHook
* IdoQueryExtensionHook
* ServiceActionsHook
* TimelineProviderHoo

Aby wykorzystać w module jeden z zaczepów należe dodać plik:

[module\_name]/run.php

Zawierający w przypadku zaczepu *GrapherHook* następującą metodę:

$this->registerHook('grapher',

'\\Icinga\\Module\\[module\_name]\\ProvidedHook\\Grapher');

Implementacja tego zaczepu musi się mieścić w:

[module\_name]\library\[module\_name]\ProvidedHook\Grapher.php

# Bibliografia

[1] Icinga documentation, docs.icinga.org

[2] Icinga www.icinga.org/resources/presentations/

[3] GNU license, www.gnu.org/licenses/old-licenses/lgpl-2.0.html

[4] Web 2.0, pl.wikipedia.org/wiki/Web\_2.0

[6] Pnp4Nagios documentation, docs.pnp4nagios.org

[7] Graphite and Whisper database, [www.graphite.wikidot.com](http://www.graphite.wikidot.com)

[9] Steven Choy, Taking your Server’s Pulse, [www.technet.microsoft.com](http://www.technet.microsoft.com)

[13] Round Robin Database, oss.oetiker.ch/rrdtool/

[10] Krzysztof Opasiak, Rozproszone monitorowanie systemów komputerowych, Wydział Elektroniki i Technik Informacyjnych, Warszawa 2014

[11] Marcin Kubik, Rozproszony system monitorowania systemów komputerowych – aplikacja mobilna na platformę Android, Wydział Elektroniki i Technik Informacyjnych, Warszawa 2014

[12] Michał Dżaman, Zdalne monitorowanie stacji roboczych, Wydział Elektroniki i Technik Informacyjnych, Warszawa 2014

1. gs.statcounter.com, listopad 2015r. [↑](#footnote-ref-1)
2. www.w3techs.com, listopad 2015r. [↑](#footnote-ref-2)
3. ang. General Public License – licencja otwartego oprogramowania, treść znajduje się w [3]. [↑](#footnote-ref-3)
4. Web 2.0 – potoczne określenie serwisów internetowych, powstałych po 2001 r., w których działaniu podstawową rolę odgrywa treść generowana przez użytkowników danego serwisu. [4] [↑](#footnote-ref-4)
5. ang. Command Line Interface – interfejs wiersza poleceń [↑](#footnote-ref-5)
6. ang. Management Information Base – baza informacji zarządzania. [↑](#footnote-ref-6)
7. Pełna lista typów i ich atrybutów znajduje się w [1], rozdział „object-types”. [↑](#footnote-ref-7)
8. gs.statcounter.com, listopad 2015r. [↑](#footnote-ref-8)
9. Jest to struktura służąca do komunikacji międzywątkowej. [↑](#footnote-ref-9)