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 2

Opiekun pracy

dr inż. Piotr Gawkowski

Ocena:

Podpis Przewodniczącego

Komisji Egzaminu Dyplomowego

Kierunek: Informatyka

Specjalność: Inżynieria Systemów Informatycznych

Data urodzenia: 1993.08.04

Data rozpoczęcia studiów: 2012.10.01

Życiorys

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:

THESIS TITLE IN ENGLISH

Summary in English.

Keywords:

Spis treści

[1 Wstęp 5](#_Toc439519909)

[2 System Icinga 2 6](#_Toc439519910)

[2.1 Architektura 7](#_Toc439519911)

[2.1.1 Rdzeń 7](#_Toc439519912)

[2.1.2 Interfejs graficzny 8](#_Toc439519913)

[2.1.3 Przepływ danych 9](#_Toc439519914)

[2.2 Monitorowanie 10](#_Toc439519915)

[2.2.1 Wykonywanie sprawdzeń 10](#_Toc439519916)

[2.2.2 Monitorowanie zdalne 11](#_Toc439519917)

[2.3 Konfiguracja 13](#_Toc439519918)

[2.4 Dodatki graficzne 15](#_Toc439519919)

[3 Monitorowanie systemu Windows 17](#_Toc439519920)

[3.1 Liczniki wydajności 17](#_Toc439519921)

[3.2 Programy współpracujące z systemem Icinga 2 20](#_Toc439519922)

[3.2.1 NSClient++ 20](#_Toc439519923)

[Bibliografia 21](#_Toc439519924)

# Wstęp

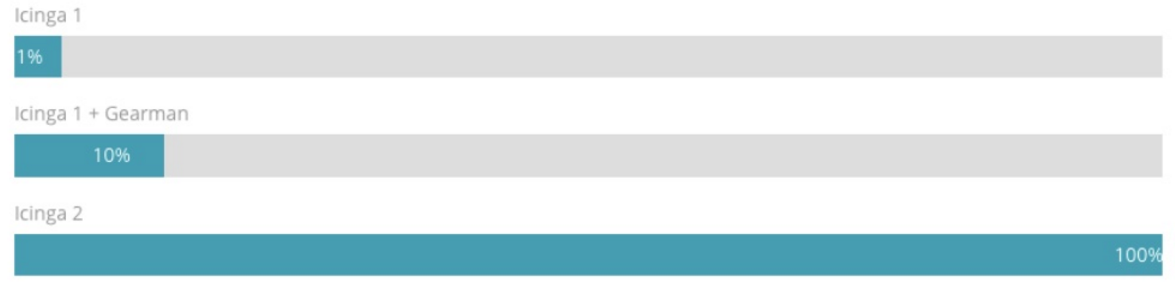
# System Icinga 2

System monitorujący Icinga powstał w 2009 r. jak 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[[1]](#footnote-1) 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[[2]](#footnote-2). Jednak główna koncepcja monitorowania pozostała zachowana, a twórcy oprogramowania zapewnili 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 oraz doprowadzenie 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.

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

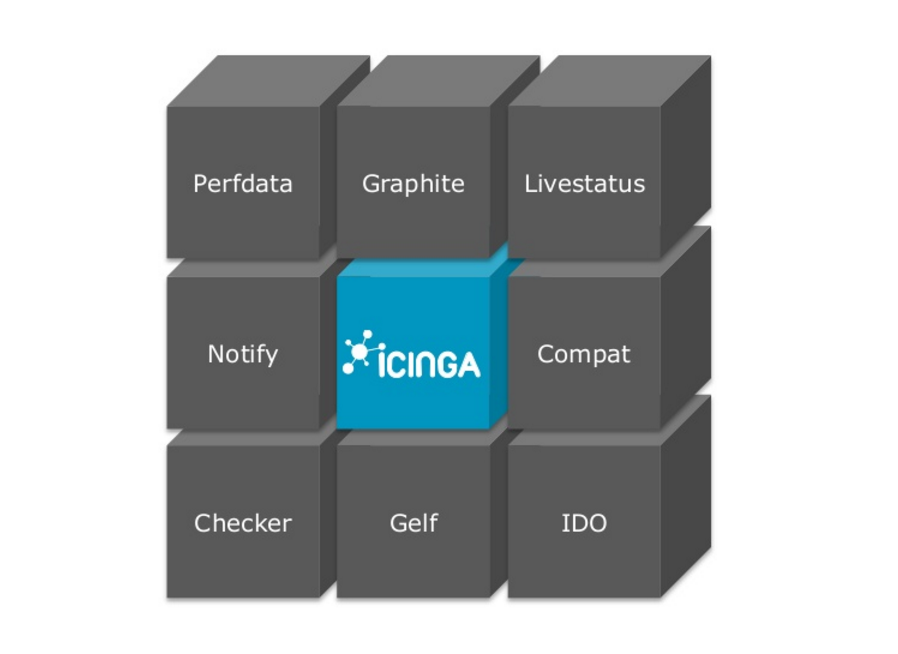
## Architektura

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 C++ 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 widać na powyższych opisach, 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[[3]](#footnote-3) dostarczone przez system Icinga.

### Interfejs graficzny

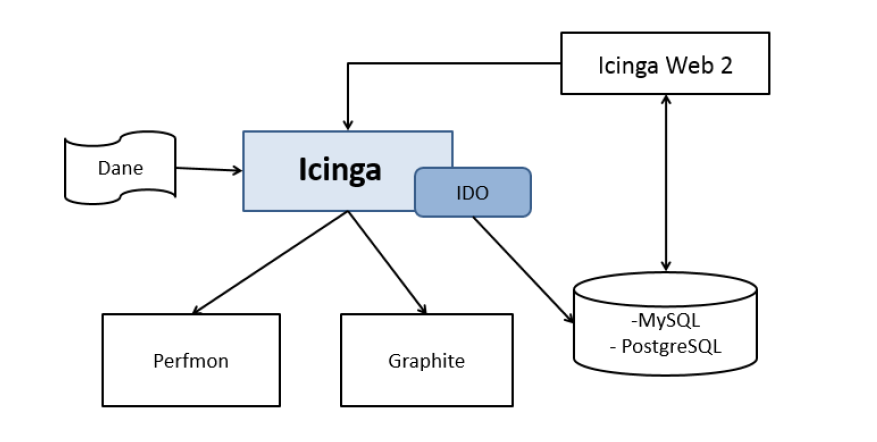
Icinga Web, nazywana interfejsem graficznym, jest w pełni funkcjonalną i niezależną aplikacją internetową (webową). Rozwój technologii internetowych oraz urządzeń mobilnych sprawił, że konieczne się stało również ulepszenie tego elementu systemu Icinga 2. Starsza wersja była mało przyjazna zarówno dla 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 Zend Framework. Użycie takich technik programistycznych jak AJAX, umożliwiło także przesyłanie danych bez odświeżania całej strony, co zmniejszyło ilość przesyłanych 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. Umożliwia 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 aby dodać własny komponent, 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 żądań wykonania komend udostępnianych przez rdzeń Icinga 2. Służy do tego plik komend zewnętrznych, który w rzeczywistości jest potokiem nazwanym. Jest to mechanizm służący do komunikacji 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 przez rdzeń. 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.

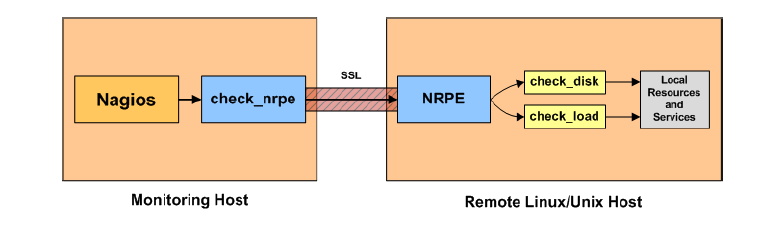
### Monitorowanie zdalne

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, wykorzystujący do przesyłania informacji 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[[4]](#footnote-4). Jak wynika z powyższego opis, 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 razem 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.

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.

**Rys. 2.4** Schemat działania dodatku NRPE

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 braku 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 tego dodatku NSCAv2, który korzysta również z nowego protokołu przesyłania danych. To rozwiązanie eliminuje opisane niedoskonałości NSCA i wprowadza kilka nowych możliwości. Nowy 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 dodania modułów przetwarzających otrzymane dane. Pozwolą 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 Icinga 2 w zonie (master). W ramach rdzenia typu master musi działać moduł IDO, który zapisuje spływające do niego dane 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.

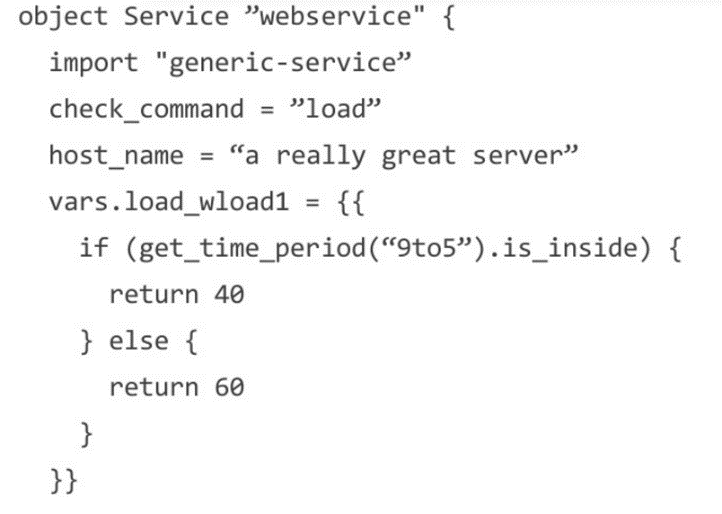
Główną zaletą klastrów jest zapewnienie wysokiej dostępności systemu monitorowania. W przypadku awarii serwera, bazy danych lub któregoś z modułów działających w ramach głównej instancji w zonie, inny rdzeń automatycznie przejmuje jego zadania i staje się węzłem typu master.

## 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)[[5]](#footnote-5). 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

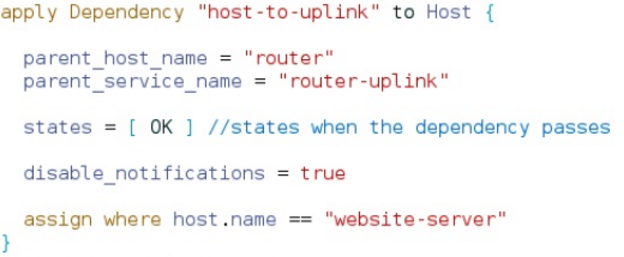
**Tabela 2.3** Przykładowe typy obiektów

|  |  |
| --- | --- |
| **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).

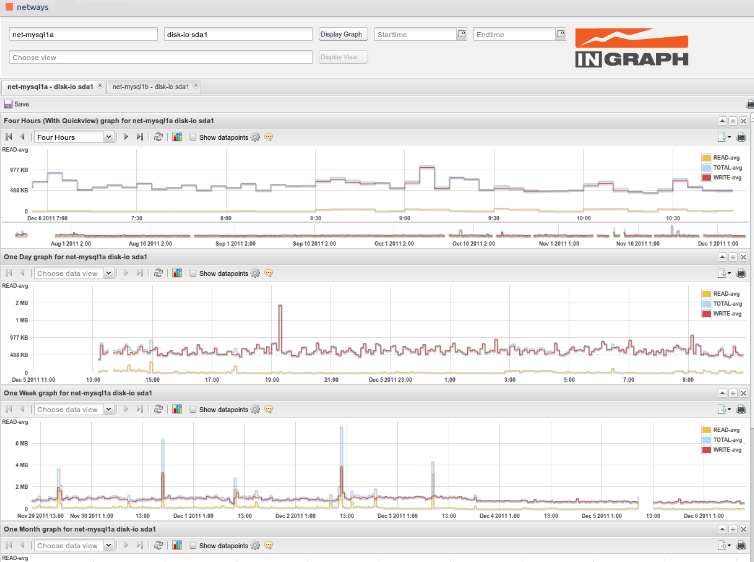
Ważną funkcją systemu monitorującego jest informowanie użytkowników o wystąpieniu sytuacji krytycznej. Sposób, w jaki się to odbywa, określają obiekty typu Notification. Są one przypisane do usług lub urządzeń oraz zawierają informację w jakich stanach oraz jak należy wysłać powiadomienie. Icinga 2 umożliwia także zdefiniowanie eskalacji powiadomień, czyli sposobu w jaki będą wysyłane kolejne wiadomości do administratorów systemu w przypadku braku reakcji na zaistniałe zdarzenie. Aby zapobiec wysyłanie 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

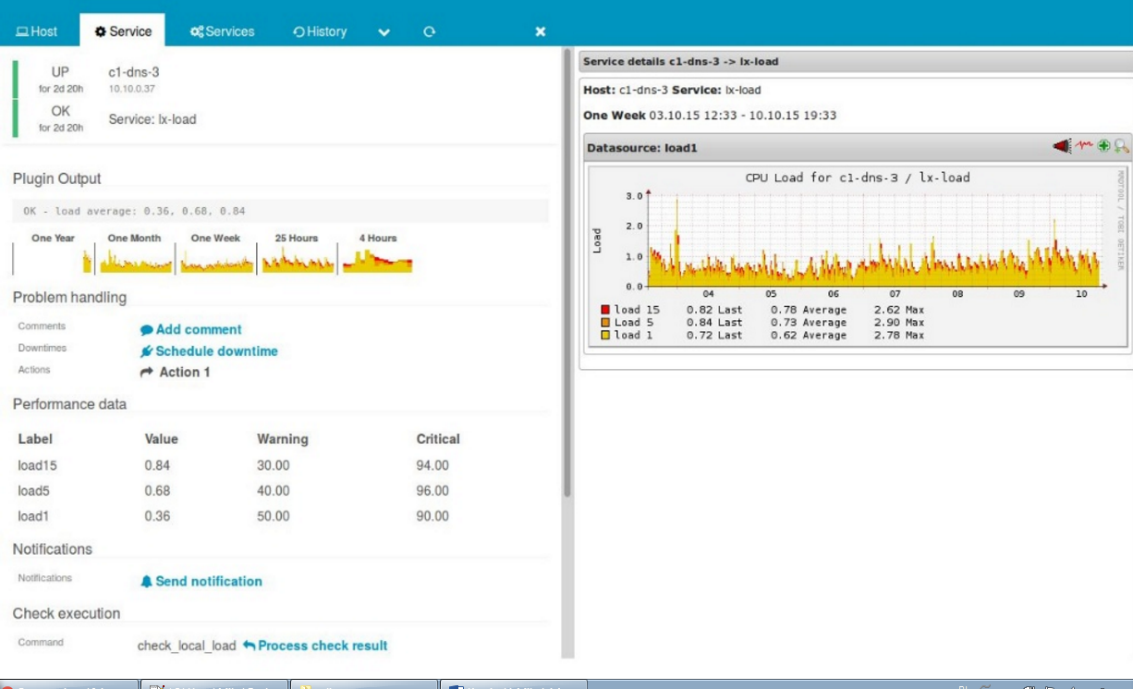
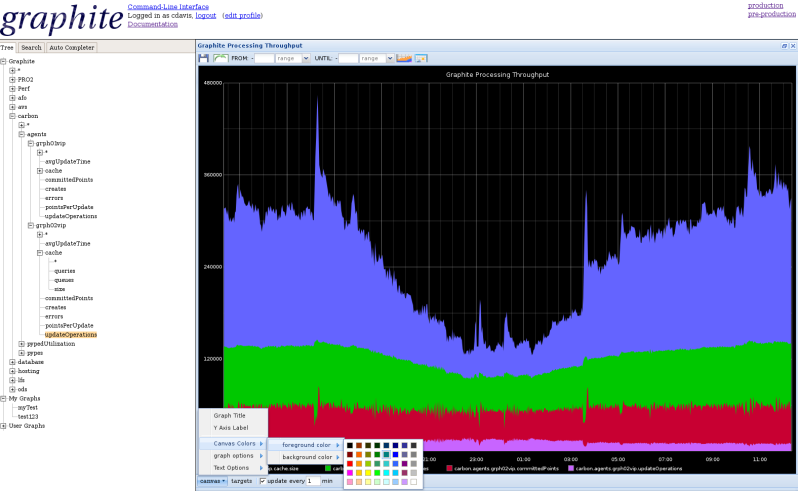


## Dodatki graficzne

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 Graphte

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 nowa baza danych Whisper.

**Rys. 2.10** Interfejs programu Graphite

**Rys. 2.9** Moduł Pnp4Nagios w interfejsie Icinga Web 2

# 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%[[6]](#footnote-6) 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.

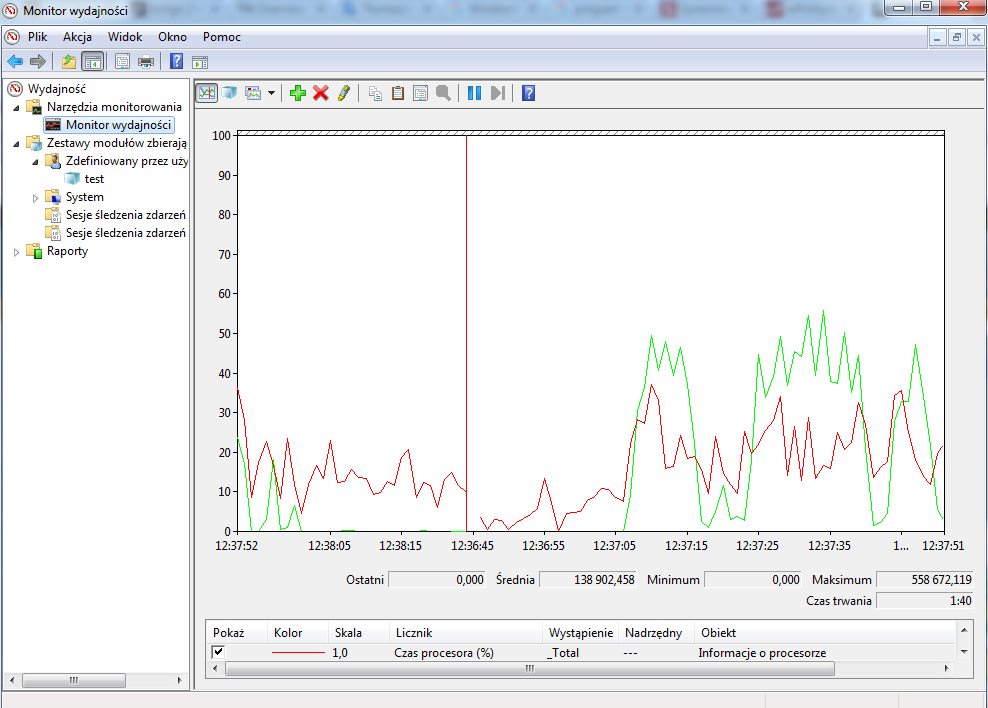
Listę wszystkich dostępnych w danym systemie liczników można sprawdzić poprzez wiersz poleceń używając komendy typeperf z odpowiednimi argumentami. Pozwala ona również na cykliczne sprawdzanie wartości liczników w określonych odstępach czasu oraz eksportowanie otrzymanych wartości do plików tekstowych, 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 sprawdzane 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 najważniejsze liczniki pozwalające monitorować te krytyczne zasoby. Przed znakiem „\” znajduje 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 trancmisji 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 monitorowanych liczników.

Monitor wydajności daje także możliwość przeprowadzenia diagnostyki systemu, której efektem jest wygenerowanie raportu mówiącego o stanie systemu. Aby uruchomić diagnostykę najpierw trzeba 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 zestawów spełniających specyficzne wymagania użytkownika. Ponadto w ramach konfiguracji takiego zestawu, można określić czas trwania oraz lokalizację oraz nazwę docelowego katalogu. Ten katalog zawiera wszystkie pliki, które są 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 danych z innych aplikacji. 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 możliwości 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 działa on na maszynach zdalnych, aby umożliwić przesyłanie danych do tych systemów trzeba użyć programów pośredniczących. NSClient++ może komunikować się z oboma dodatkami do Icingi 2: NSCA oraz NRPE, co umożliwia wykonywanie sprawdzeń aktywnych i pasywnych.

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 korzystać 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 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 także, że ciężko jest także korzystać z innych wspieranych rozwiązań oraz dodawać własne moduły. 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.

# Bibliografia

[1] docs.icinga.org

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

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

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

[5] [www.assets.nagios.com/downloads/nagioscore/docs/](http://www.assets.nagios.com/downloads/nagioscore/docs/)

[6] docs.pnp4nagios.org

[7] [www.graphite.wikidot.com](http://www.graphite.wikidot.com)

[8] [www.msdn.microsoft.com](http://www.msdn.microsoft.com)

[9] Steven Choy „Taking your Server’s Pulse” www.technet.microsoft.com

1. ang. General Public License – licencja otwartego oprogramowania, treść znajduje się w [3]. [↑](#footnote-ref-1)
2. 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-2)
3. ang. Command Line Interface – interfejs wiersza poleceń [↑](#footnote-ref-3)
4. ang. Management Information Base – baza informacji zarządzania. [↑](#footnote-ref-4)
5. Pełna lista typów i ich atrybutów znajduje się w [1], rozdział „object-types”. [↑](#footnote-ref-5)
6. www.netmarketshare.com [↑](#footnote-ref-6)